Navigasjon på kulejord

Gyro og akselerometer

Kenneth Holterhuset Johansen

Masteroppgave
i elektronikk og datateknologi

Fysisk institutt
Det matematiske- naturvitenskaplige fakultet

UNIVERSITETET I OSLO

06.06.2011

Sammendrag

MEMS sensorer, av type gyro og akselerometer, har gjennom årene blitt et stort forskningsfelt med en rekke kommersielle suksesser. I den forbindelse er det utarbeidet en oppgave for å utvikle et fysisk- og mekanisert system for et fly, bestående av navigasjonslikninger for lavkost treghetsnavigasjon på kulejord. Rammene som er brukt for å beskrive tilstandsvektorene til flyet er definert ut fra en treghetsramme, jordramme, tangentramme og body ramme. Systemene som er utviklet i oppgaven er med på å beskrive deler av et treghetsnavigasjonssystem. Det fysiske systemet representerer en banegenerator og er konstruert til å gi ut sanne tilstander som beskriver flyets posisjon, hastighet og orientering.

Pådraget til treghetsnavigasjonssystemet er beregnet på grunnlag av det fysiske systemet og gir spesifikk kraft og vinkelhastighet.

Differensiallikninger anvendes for å beskrive det mekaniserte systemet på grunnlag av teori og beregnede estimater. Bruk av integrasjonsalgoritmer fra Euler's og Heun's metode, resulterer i flyets navigasjonstilstander. Dette kan diskuteres og sammenliknes med tilstandene fra det fysiske systemet. Resultatet viser til at et komplett treghetsnavigasjonssystem vil navigere flyet bedre.

Forord

Denne oppgaven er formulert og utgitt av Universitetet på Kjeller (UNIK) i vårsemesteret 2011. Oppgaven er en avsluttende del av masterutdanningen i elektronikk og datateknologi ved Universitetet i Oslo (UiO), utført ved UNIK. Prosjektet gjennomføres av undertegnende, som vil takke intern veileder og faglærer Oddvar Hallingstad for stor hjelp og verdifull kunnskap som har gjort oppgaven veldig lærerik. En takk rettes også til venner og familie som har vært støttende gjennom perioden.

Kjeller, 6. juni 2011

_____

Kenneth Holterhuset Johansen

Universitetet i Oslo

Innholdsfortegnelse

Figurer

Tabeller

Definisjoner og forkortelser

UiO	Universitetet i Oslo
UNIK	Universitetssenteret på Kjeller
MEMS	Mikro Elekro-Mekaniske Systemer
TNS	Treghetsnavigasjonssystem
NAV	Navigasjonslikninger
RKM	Rotasjonskosinmatrise
KTM	Koordinattransformasjonsmatrise
CCW	Counter Clockwise
CW	Clockwise
MATLAB	PC program: Algoritmisk arbeidsverktøy
{i}	Treghetsramme (ECEF - Earth-Centered Earth-fixed ramme)
{e}	(ECI - Earth-Centered-Inertial ramme)
{t}	Tangentramme
{b}	BODY-fikset ramme (BODY)

Matematiske symboler

φ	Fi
θ	Theta
Ψ	Psi
Δ	Delta
π	Pi
α	Alfa
ω	Omega

1 Innledning

1.1 Generelt

Oppgaven er blitt utgitt av UNIK med intern og ekstern veileder Oddvar Hallingstad. I denne forbindelse vil faget UNIK 4540 – Matematisk modellering av dynamiske systemer være med på å påvirke masteroppgaven å støtte opp mye av den matematiske teorien i rapporten. Faget har vært en del av masterutdanningen som er blitt undervist på Kjeller hvor veileder har hvert forleser.

Mye tid av oppgaven har blitt brukt på å tilegne seg mye ny kunnskap. Dette gjelder spesielt for å realisere en banegenerator som fungerer optimalt og kan brukes til å beskrive et fysisk system. Mye matematikk og studering av ny teori, samt deler av tidligere pensum, har ført til at oppgaven har vært veldig lærerik. Dette har gitt nærmere innsikt innenfor et spesielt fagområde og tas med som god erfaring videre.

Oppgaven ble utdelt 31. januar 2011 med en tidsramme på 18 uker, som tilsier et dokument arbeid leveres innen 6. juni 2011.

Til opplysning er følgende notasjon: $^{\mathbf{x}}$ benyttet for å henvise til beskrivelse av ord og uttrykk. Følgende notasjon: $^{\mathbf{[x]}}$ er brukt for å henvise til referanser.

1.2 Målsetting for oppgaven

Oppgaven handler om å sette seg inn i hvilken rolle lavkost MEMS sensorer, i form av gyro og akselerometre, har i forhold til navigasjon for et fly. Disse sensorene kan forbindes med et treghetsnavigasjonssystem som delvis består av et fysisk- og mekanisert system. Målingsparametere fra sensorene omgjøres til beregnede verdier som skal utvikles fra det fysiske systemet.

Representasjonen av det fysiske systemet vil være å realisere en deterministisk banegenerator med bruk av punkter og vektorer beskrevet i et 2-dimensjonalt affint rom. Navigeringen til det sanne systemet skal skje med tanke på kulejord, og representere flyet i ulike referanserammer som kan beskrive ulike tilstandsvektorer.

Disse tilstandene skal brukes videre for å beregne et pådrag inn på TNS og videreutvikle navigasjonslikninger av differensial form.

Navigasjonslikningene skal gi ut informasjon om flyets posisjon, hastighet og orientering basert på matematisk og teoretisk grunnlag. Det samme for det fysiske systemet.

Sluttresultatet for realisert og utarbeidet fysisk system, skal kunne sammenliknes med det beregnede mekaniserte systemet. Er resultatene like i forhold til hverandre og kan tilstandene til navigasjonslikningene brukes til å navigere et fly like optimalt som det fysiske? Reflektering og beskrivelse av resultatet skal til slutt oppsummeres til en konklusjon om hva som er oppnådd og hva som ble av resultater.

1.3 Kapittelinndeling

Kapittel 2: Matematisk grunnlag

I dette kapittelet beskrives ulike matematiske grunnlag for oppgaven tatt opp i forskjellige deler.

Kapittel 3: Banegenerator

I dette kapittelet beskrives først en teoretisk og matematisk fremgangsmåte for å realisere en banegenerator. Tilslutt beskrives funksjoner for programmeringen som er utført i form av figurer.

Kapittel 4: Treghetsnavigasjon (TNS)

I dette kapittelet beskrives et treghetsnavigasjonssystem hvor det fysiske systemet blir representert av banegeneratoren, og et mekanisert system er utledet med teori og matematiske formler. I tillegg blir funksjonaliteten og rollen til MEMS gyro og akselerometer, i forhold til et TNS, bli beskrevet.

Kapittel 5: Simuleringsresultater

I dette kapittelet blir resultatene som er oppnådd fremstilt i grafer og figurer.

Kapittel 6: Konklusjon

I dette kapittelet blir oppnådde resultater oppsummert og konkludert.

Kapittel 7: Videre arbeid

I dette kapittelet blir det tatt opp forslag til videre arbeid for oppgaven, basert på oppnådde resultater.

2 Matematisk grunnlag

I denne delen blir ulike matematiske grunnlag for oppgaven tatt opp i forskjellige deler. Dette viser begrunnelser for valg av metoder og framgang i matematikken og hvorfor bestemte algoritmer og beregninger er blitt valgt til løsning. Definerte likninger og teori vil ofte bli referert til ved bruk av dette grunnlaget.

2.1 Vektorer og matriser

Vektor- og matriseberegninger brukes generelt for å finne avstander, retninger og oppførselen til et legeme når man ser bort fra fysiske krefter som har en påvirkning. Om legemet er i lufta, på bakken eller under vann har ingen betydning, samme matematiske formler og beskrivelser kan benyttes over store deler hvor fysiske lover er relevant. Del 2.1.1 omhandler vektorregning og definisjoner som gir en retningsvinkel på hvilke formler som er benyttet i oppgaven. Det samme gjelder for matriser.

2.1.1 Vektorer

En type vektor blir ofte definert i et vektorrom ($\mathcal{V}$) eller et affint rom ($\mathcal{A}$). Notasjonen for de sentrale begrepene i et vektorrom og affint rom innenfor matematikken, samt referanse- og treghetsrom, er gitt i tabellen[1]:

Rom	**Rammer**	**Andre navn på rammer**
$\mathcal{V}$: vektorrom	$F_a^{\mathcal{V}} = \left\{\vec{a}_1,\ \vec{a}_2, \vec{a}_3\right\}$: ramme a i vektorrom $\mathcal{V}$ med basisvektorer $\vec{a}_i$	basis, basissystem, basisvektorsett k.s.

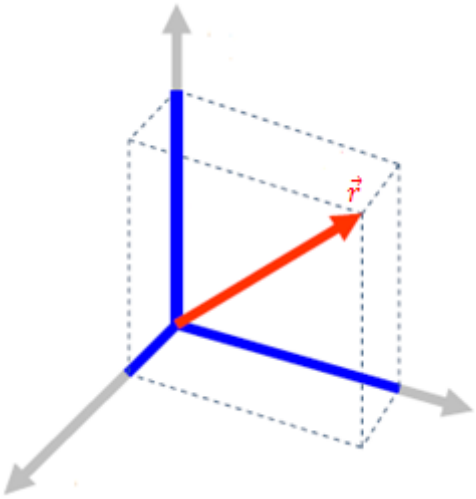
Tabell 2.1.1: Notasjon for vektorrom

Rom	**Rammer**	**Andre navn på rammer**
$\mathcal{A}$: affint rom	$F_a^{\mathcal{A}} = \{O_a, \vec{a}_1, \vec{a}_2, \vec{a}_3\}$ $= \{O_a, \vec{x}_a, \vec{y}_a, \vec{z}_a\}$ ramme a i affint rom A med Origo O_a og basisvektorer $\vec{a}_i$	k.s.

Tabell 2.1.2: Notasjon for affint rom

Det generelle vektor- og affine rommet kan bestå av flere vektorelementer, og kan tjene som modell for et fysisk tredimensjonalt rom, hvor klassisk mekanikk er formulert.

Når man definerer et referanserom brukes ofte punkter som viser hvordan et stivt legeme beveger seg relativt i referanserommet. For å vise forflytning, hastighet og akselerasjon kan piler brukes som illustrasjon. For stive legemer kan basisvektorer innføres i et vektorrom for å vise orienteringen. Dette forklarer hvordan et affint rom henger sammen med et vektorrom. Et eksempel på vektorrom kan være geometriske objekter med retninger, lengde (piler) og n-tupler av tall samlet i en kolonnematrise.



Figur 2.1.1: Eksempel på retningsvektor i et referanserom[2]

En vektor $\vec{r}$ kan beskrives ved sin størrelsesorden og retning, derfor trenger den ikke nødvendigvis være avhengig av en bestemt koordinatramme. I et kartesisk koordinatsystem hvor p er representert sammen med den ortogonale enhetsvektoren $\vec{p}_i$, er vektor $\vec{r}$ angitt som en lineær kombinasjon av enhetsvektoren $\vec{p}_i$ i rommet $\mathbb{R}^n$ ved[3]

$$\vec{r} = \sum_{i=1}^{n} r_i^p \, \vec{p}_i \tag{2.1}$$

Hvor

$$r_i^p = \langle \vec{r}, \vec{p}_i^* \rangle \tag{2.2}$$

Fra denne beregningen kan r_i^p, hvor i representerer antall vektorer i rommet, angis som en kolonnevektor eller algebraisk vektor

$$\vec{r} = \begin{bmatrix} r_1 \\ \vdots \\ r_2 \end{bmatrix} \tag{2.3}$$

For å finne vektorlengden til et referanserom skrives

$$|\vec{r}| = \sqrt{r_1^2 + \cdots + r_i^2} \tag{2.4}$$

Generaliseringen av en vektorlengde til et referanserom kalles en *norm*[1] og skrives som $\|\vec{r}\|$. Lengden av en geometrisk vektor svarer til den *euklidiske*[2] normen, også kalt 2-normen. På grunn av denne sammenhengen ser en ofte bruk av symbolet $\|\vec{r}\|$ også som betegnelse på den (eukidiske) vektorlengden.

Generelt kan et kryssprodukt[6] mellom to geometriske vektorer beregnes på to forskjellige måter. En metode som ofte blir brukt kan beskrives med to vektorer *r* og *v* og er definert som

$$\vec{r} \times \vec{v} = |\vec{r}||\vec{v}|sin\theta \tag{2.5}$$

Den andre metoden kan uttrykke vektorene i et $\mathbb{R}^2$ rom i form av matrise, og beregnes fra

$$\overrightarrow{rv} = \begin{bmatrix} r_1 & r_2 \\ v_1 & v_2 \end{bmatrix} \tag{2.6}$$

til

$$\vec{r} \times \vec{v} = (r_1 * v_2) - (r_2 * v_1) \tag{2.7}$$

For et $\mathbb{R}^3$ rom

$$\vec{\omega} = \vec{r} \times \vec{v} = \begin{bmatrix} \vec{\imath} & \vec{\jmath} & \vec{k} \\ r_1 & r_2 & r_3 \\ v_1 & v_2 & v_3 \end{bmatrix} \tag{2.8}$$

[1] En norm er i matematikk en funksjon som tilordner seg en lengde til enhver vektor i et vektorrom. Lengden er en reell skalar og vil være positiv for alle vektorer, bortsett fra for nullveltoren, som har lengde lik null[4].
[2] Euklid er en gresk matematiker av Alexandira og blir veldig ofte referert til under mange matematiske omstendigheter på grunn av sin historiske bakgrunn[5].

Matrise representasjonen av $\vec{\omega}$ kan defineres videre til en skjevsymmetrisk matrise som brukes for eksempel til å regne ut spesifikk kraft og navigasjonslikninger. Vektoren $\vec{\omega}$ beskrives som en vinkelhastighet innenfor kinematikk.

2.1.2 Skjevsymmetrisk matrise

Ved å innføre en skjevsymmetrisk form[7] av vektor $\vec{\omega}$ kan matrisen skrives som $S(\underline{\omega}^p)$. Denne formen kan også skrives som en "$\vec{\omega}\times$"-operator, og i en ortonormal basis, $\{\vec{p}_i\}$, er den gitt ved:

$$S(\underline{\omega}^p) = \underline{\omega}^p \times = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix} \tag{2.9}$$

2.1.3 Rotasjonskosinmatrise (RKM)

Retningskosinmatrisen kan brukes for å finne en overgang fra to eller flere forskjellige referanserammer i et helt system. Et eksempel kan forklares bedre ved å se for seg en ramme a og en ramme b. En rotasjonsoperator kan dermed beskrives som en lineær operator $R_a^b: \mathcal{V} \to \mathcal{V}$, definert ved [8]:

$$\vec{b}_i = R_a^b \vec{a}_i \tag{2.10}$$

hvor

$$i \in \{1, 2, 3\}$$

Hvor basisvektorsettet $\{\vec{a}_i\}$ samlet roteres til en ny stilling slik at $\vec{a}_i \to \vec{b}_i$.
Dyaderepresentasjonen av R_a^b er da:

$$R_a^b = \vec{b}_1\vec{a}_1^* + \vec{b}_2\vec{a}_2^* + \vec{b}_3\vec{a}_3^* \tag{2.11}$$

Fordi

$$R_a^b \vec{a}_i = \vec{b}_i; \quad i = 1, 2, 3 \tag{2.12}$$

Teorem A.7 fra referanse: [8] viser dermed til at matriserepresentasjonen av rotasjonsoperatoren R_a^b i ramme *a* og *b* er

$$[R_a^b]^a = [R_a^b]^b = C_a^b \tag{2.13}$$

For å si dette må R_a^b operatoren kunne tilfredsstille

$C_b^a \underline{y}^b = [R_a^b]^a \underline{x}^a$	$\underline{y}^b = [R_a^b]^b C_a^b \underline{x}^a$
$\underline{x}^a = C_a^b [R_a^b]^a \underline{x}^a$	$\underline{x}^a = [R_a^b]^b C_a^b \underline{x}^a$
$(I - C_a^b [R_a^b]^a)\underline{x}^a = \underline{0}\ for\ alle\ \underline{x}^a$	$(I - [R_a^b]^b C_a^b)\underline{x}^a = \underline{0}\ for\ alle\ \underline{x}^a$
$C_a^b [R_a^b]^a = I$	$[R_a^b]^b C_a^b = I$
$[R_a^b]^a = C_b^a$	$[R_a^b]^b = C_b^a$

Det vil si at matriserepresentasjonen av R_a^b i *a*- og *b*-rammene er den samme.

Tolkningen eller bruk av RKM C_b^a kan summeres opp i tre punkter:

1. C_b^a: er en *koordinattransformasjonsmatrise* (KTM) som gir sammenhengen mellom koordinatene i rammene *b* og *a* osv, mellom de algebraiske vektorene $\underline{r}^b$ og $\underline{r}^a$:

$$C_b^a : \underline{r}^b \rightarrow \underline{r}^a = C_b^a \underline{r}^b \tag{2.14}$$

2. C_b^a: er en *stillingsmatrise* (attitude matrix) som gir stillingen (orienteringen) av ramme *b* i ramme *a*: dvs kolonnene i C_b^a er en kolonnerepresentasjon av *b*-basisen i *a*-basisen.

$$C_b^a = [b_1^a, b_2^a, b_3^a] \tag{2.15}$$

3. $C_b^a = [R_a^b]^a = [R_a^b]^b$ er en rotasjonsmatrise som kan brukes til å rotere en vektor i *a*- eller *b*-ramma:

$$\vec{r}_2 = R_a^b \vec{r}_1 \tag{2.16}$$

$$\underline{r_2}^a = C_b^a \underline{r_1}^a \tag{2.17}$$

$$\underline{r_2}^b = C_b^a \underline{r_1}^b \tag{2.18}$$

Hvilket navn som brukes på C_b^a er som vist avhengig av hvilken tolkning som er aktuell. RKM avspeiler den matematiske beregningen av elementene i C_b^a, og kan brukes i forbindelse

med alle tolkningene. Tolkning en og to kalles også for en *passiv* tolkning fordi det er koordinatsystemet som endres. Tolkning tre kalles *aktiv* fordi det er vektoren som endres.

Rotasjonsmatriser brukes også ofte i matematisk sammenheng til å uttrykke Eulervinkler.

2.1.4 Eulervinkler

Eulervinkler[3] er en ganske vanlig metode å benytte for å beskrive stillingsformen til et legeme. I en bestemt ramme som er definert, roteres tre forskjellige koordinatakser med vinklene θ_1,θ_2 og θ_3 som er definert i hver sin egen rotasjonsmatrise. Vinkelen θ_3 roteres om z-aksen, θ_2 roteres om y-aksen og til slutt roteres θ_1 om x-asken. Disse tre sekvensene av rotasjoner blir kalt zyx-konvensjonen, avhengig av hvilken rekkefølge man roterer om. Aksene står ortonormale på hverandre og oppfyller høyrehåndsregelen. Ved å multiplisere disse tre rotasjonene kan man finne rotasjonssekvensen til et stivt legeme, og i tillegg finne rotasjonen mellom to forskjellige rammer[9].

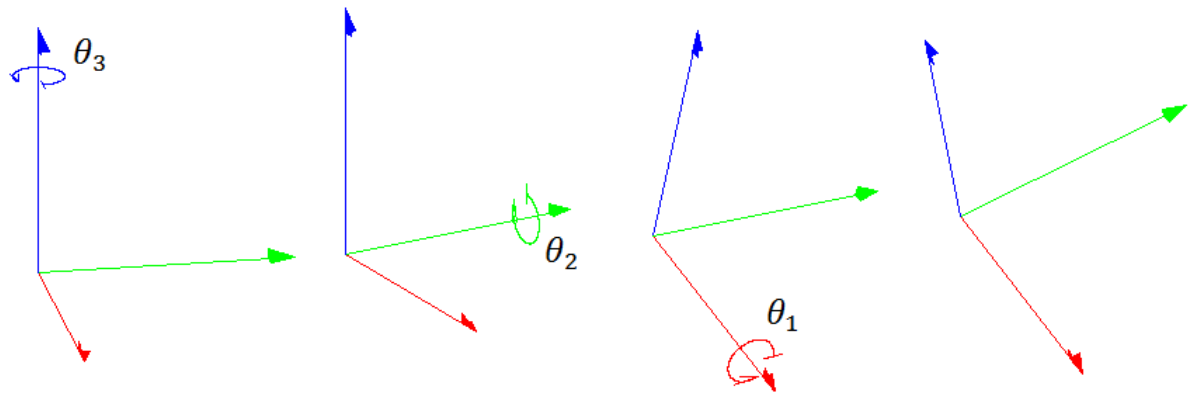
$$R_a^b(\theta_3, \theta_2, \theta_1) = R\ (\theta_3)R(\theta_2)R(\theta_1) \tag{2.19}$$

Er det faste eller nye akser (Eulervinkler), vil det til slutt gi samme endelige stilling for begge tilfeller, men med forskjellig rekkefølge for rotasjon.

$${}^E R_a^b(\theta_3, \theta_2, \theta_1) = {}^F R_a^b(\theta_1, \theta_2, \theta_3) \tag{2.20}$$

Etter denne betingelsen kan man benytte tolv forskjellige definisjoner av RKM. Figuren 2.1.2 under illustrerer en sekvens med tre rotasjoner 3-2-1 (Z-Y-X – aksene), og viser hvordan et stivt legeme kan endre seg med tiden. Ved bruk av rotasjon om 1-2-3 eller 3-2-1, konkluderes det med at man ender opp i samme stilling som ønsket.

[3] Eulervinkler er tre vinkler introdusert av Leonhard Euler og brukes til å beskrive orienteringen i et stivt legeme.



Figur 2.1.2: Rotasjonssekvens 3-2-1 (Eulervinklene)[10]

De tre generelle rotasjonsmatrisene til Euler defineres som:

$$R(\theta_3, z) = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 \\ s\theta_3 & c\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.21}$$

$$R(\theta_2, y) = \begin{bmatrix} c\theta_2 & 0 & s\theta_2 \\ 0 & 1 & 0 \\ -s\theta_2 & 0 & c\theta_2 \end{bmatrix} \tag{2.22}$$

$$R(\theta_1, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta_1 & -s\theta_1 \\ 0 & s\theta_1 & c\theta_1 \end{bmatrix} \tag{2.23}$$

Følgende forkortelse blir brukt for notasjonen til de tre generelle rotasjonsmatrisene:

$$c(\cdot) = cos(\cdot)$$
$$s(\cdot) = sin(\cdot)$$

Hvor til sammen hele rotasjonsmatrisen $R_a^b(\theta_3, \theta_2, \theta_1)$ kan skrives slik:

$$R_a^b(\theta_3, \theta_2, \theta_1) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c\theta_3 c\theta_2 & c\theta_3 s\theta_2 s\theta_1 - s\theta_3 c\theta_1 & c\theta_3 s\theta_2 c\theta_1 + s\theta_3 s\theta_1 \\ s\theta_3 c\theta_2 & s\theta_3 s\theta_2 s\theta_1 + c\theta_3 c\varphi & s\theta_3 s\theta_2 c\theta_1 - c\theta_3 s\theta_1 \\ -s\theta_2 & c\theta_2 s\theta_1 & c\theta_2 c\theta_1 \end{bmatrix}$$

(2.24)

Notasjon for θ-vinklene i treghetssammenheng er gitt i tabellen under:

Vinkel	Beskrivelse
θ_1	Rullvinkel, bruker ofte φ
θ_2	Elevasjonsvinkel (pitch/stamp), bruker ofte θ
θ_3	Asimutvinkel (yaw/kurs), bruker ofte ψ

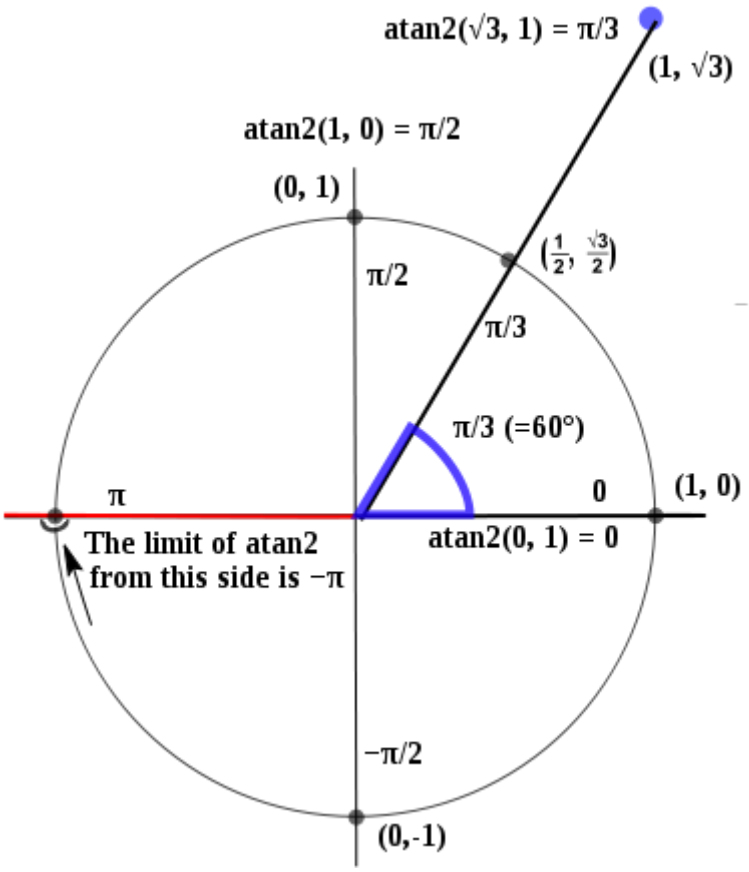
Tabell 2.1.3: Treghetssammenheng av Eulervinklene

Likning (2.24) fremstiller 3-2-1 Eulervinklene, og etterfulgt av dette oppfylles to hovedgrupper som inntreffer ved rotasjon.

1. Rotasjoner om tre forskjellige akser får vi singularitet for $\theta = \pm 90^{\circ}$.

2. Tre rotasjoner om to forskjellige akser får vi singulæritet for $\theta = 0^{\circ}$ og $\theta = 180^{\circ}$.

Dette kalles for det *inverse problem*[11] og løses ved å se på hvert element i R_a^b. Ved å bruke *atan2* i MATLAB, returneres verdier av vinklene i riktig kvadrant for hvert element, løses singularitetsproblemet.



Figur 2.1.3: atan2[12]

I figuren 2.1.3 er det skissert opp grenser og punkter i en enhetssirkel for å vise bedre forståelse for hvordan atan2 kan bidra til å forenkle vinkelproblemer.

2.1.5 Kinematisk problem

I enkelte tilfeller for treghetsnavigasjon benytter man seg av det *kinematiske problem* for 3-2-1 Eulervinkler for å beskrive orienteringen til et fly. Ved dette fremstilles en differensiallikning for RKM, gitt at vinkelhastigheten $\underline{\omega}_a^b(t)$ er definert eller målt, og har kjennskap til $R_a^b(t_0)$. Hvis RKM er representert med hensyn på Eulervinklene, ønskes det å skrive differensiallikningen direkte i Eulervinklene. Dette oppfylles hvis vinkelhastighet $\underline{\omega}_a^{ba}$ er definert som en vektor på følgende form:

$$\underline{\omega}_a^{ba} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.25}$$

Ved dette finner man den deriverte av θ_1, θ_2 og θ_3 beskrevet av ${}^{E}R_a^b(\theta_3, \theta_2, \theta_1)$

$$\dot{\theta} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} 1 & s\theta_1 t\theta_2 & c\theta_1 t\theta_2 \\ 0 & c\theta_1 & -s\theta_1 \\ 0 & \dfrac{s\theta_1}{c\theta_2} & \dfrac{c\theta_1}{c\theta_2} \end{bmatrix} * \underline{\omega}_a^{ba} \tag{2.26}$$

Hvor

$$t(\cdot) = tan(\cdot)$$

Dette bekreftes fra likning A-106 i artikkelen: [13]

2.2 Koordinatsystemer og rammer

Når man foretar måling eller beregninger av et fartøys bevegelse, refereres dette i forhold til ulike referansesystemer eller rammer. I dette tilfellet vil rammene være:

- Treghetsramme – ECI
- Jordramme – ECEF
- Tangentramme
- Legemeramme – BODY

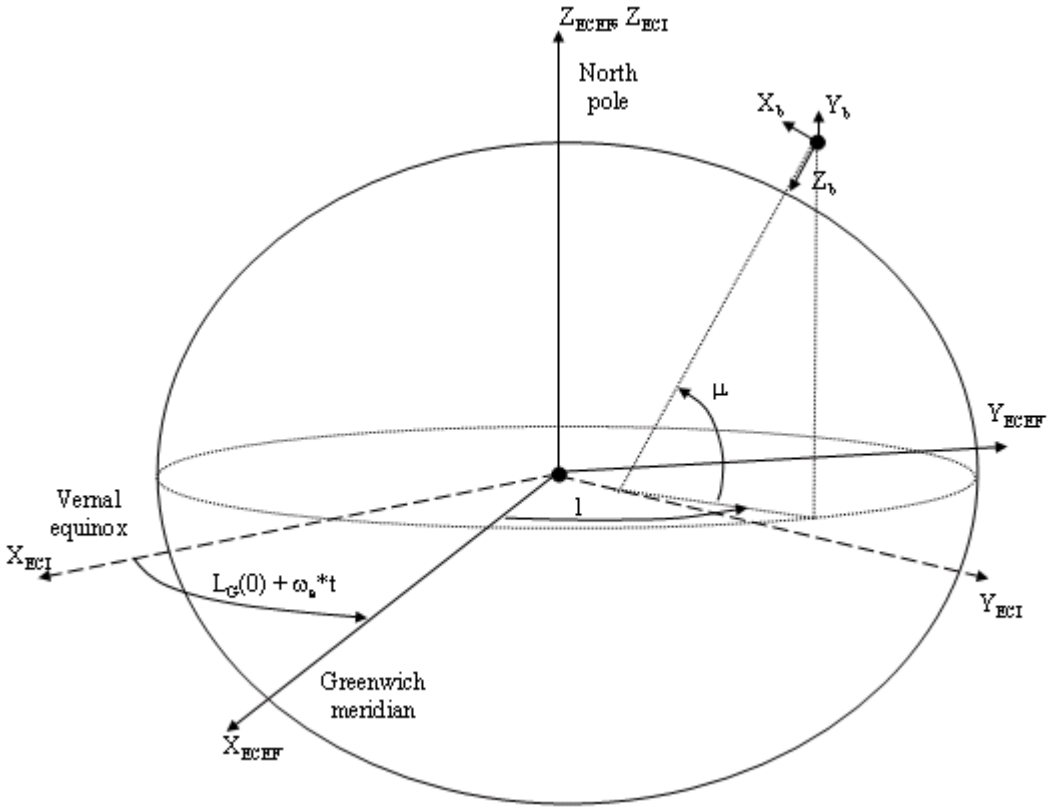
Disse defineres best med hvert sitt koordinatsystem. For et TNS er det derfor viktig å ha klart for seg hvordan et fartøys bevegelse (posisjon, hastighet, akselerasjon og orientering) uttrykkes i de enkelte koordinatsystemene, samt hvordan en dekomponerer og transformerer vektorenestørrelser mellom de ulike systemene. Dette blir kalt kinematikk.

Punktene videre vil beskrive de ulike ortonormale, høyrehåndsorienterte koordinatsystemene som er funksjonelle i henhold til navigasjon innenfor jordens atmosfære. Det blir under beskrevet jordfaste og fartøyfaste koordinatsystem, med transformasjon mellom de ulike rammene. ECI og ECEF er to rammer som kan kategoriseres som jordsentrerte (geosentriske) koordinatsystemer, og BODY ramme som fartøyfast koordinatsystem.

2.2.1 Treghetsramme – ECI

Koordinatsystemet ECI står for Earth-Centered-Inertial[14], er plassert i jordas origo O_i, referer aksene til fiksstjernene i verdensrommet og roterer ikke om jorda. ECI er en treghetsramme (inertialframe) og har notasjon {i} hvor aksene er ikke akselerert i forhold til fiksstjernene. Newtons 2.lov kan dermed beregnes i dette koordinatsystemet. Aksen $\vec{z}_i$ peker mot geografisk nordpol, mens $\vec{x}_i$ og $\vec{y}_i$-aksene danner et høyrehånds ortogonalt koordinatsystem og peker mot jordas ekvator ved tiden t_0. Akserammen kan skrives på følgende form:

$$F^{\{i\}} = \{O_i, \vec{x}_i, \vec{y}_i, \vec{z}_i\}$$



Figur 2.2.1: ECI- , ECEF- og BODY-ramme[15]

2.2.2 Jordramme – ECEF

Koordinatsystemet ECEF står for Earth-Centered Earth-fixed[16] og har samme origo som ECI, jordsenteret. Rammen har notasjon {e}, hvor $\vec{z}_e$-aksen sammenfaller om jordas ECI rotasjonsakse og $\vec{x}_e\vec{y}_e$-planet sammenfaller med ekvatorplanet med $\vec{x}_e$-akse pekende 90˚ mot krysningspunktet mellom Greenwitch-medianen og ekvator. Akse $\vec{y}_e$ bidrar til høyrehåndskoordinatsystem. Jordrammen kan dermed beskrives som:

$$F^{\{e\}} = \{O_e, \vec{x}_e, \vec{y}_e, \vec{z}_e\}$$

Forholdet mellom ECI og ECEF er jordrotasjonen, og ved tiden t_0 er de like. ECEF roterer med en vinkelhastighet $\omega_e = 7{,}292115e^{-5}\ rad/sek$ om felles z-akse, hvor tidligere fortalt ECI har sin referanse mot fiksstjernene. For de fleste fartøysanvendelser vil ECEF være en tilnærmet treghetsramme.

For å gi en forståelig posisjonsbeskrivelse av rotasjonen mellom ECEF og ECI, bruker man lengdegrader, breddegrader og høyde. Dersom vinkelen mellom $\vec{x}_e$-aksen er l_0 (inertiell lengdegrad) ved tiden t_0, kan man finne endring i lengdegrad ettersom tiden øker:

$$l = l_0 + (\omega_e * t) \tag{2.27}$$

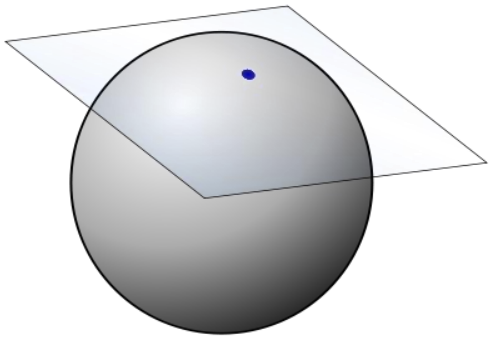
Radiusen på jordrammen måles fra jordas sentrum O_e (origo) og ut til ytterste punktet på $\vec{y}_e$-aksen, tangentrammen finner sted med 60˚ vinkling opp fra denne aksen. Jordradiusen r_e er på $6{,}370e^6$ m. For å finne representasjonen av punkter i ulike koordinatsystemer må avstanden mellom origoene oppgis fra ramme {e} til {i} hvor posisjonsvektoren $\underline{p}_e^i$ er gitt ved:

$$\underline{p}_e^i = \underline{0} \tag{2.28}$$

2.2.3 Tangentramme

Tangentrammen har notasjon {t} og består av et koordinatsystem $\vec{x}_t$, $\vec{y}_t$ og $\vec{z}_t$ som krysser jordrammen tangentielt. Hvor $\vec{x}_t$ har retning østover, $\vec{y}_t$ peker nordover og $\vec{z}_t$ peker oppover. Rammen $F^{\{t\}}$ kan beskrives på følgende måte:

$$F^{\{t\}} = \{O_t, \vec{x}_t, \vec{y}_t, \vec{z}_t\}$$

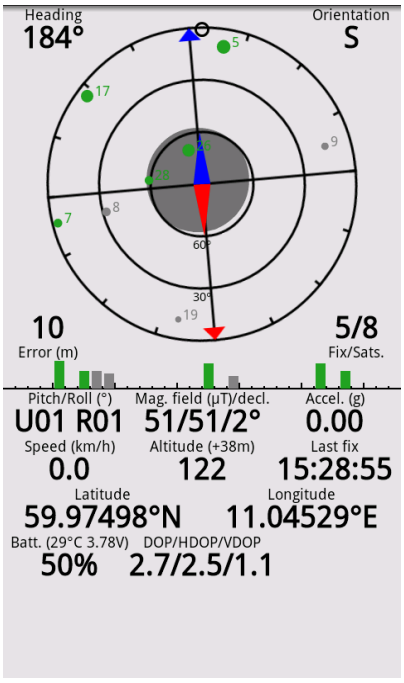


Figur 2.2.2: Tagentramme

Origo O_t sammenfaller $\vec{x}_t$ og $\vec{z}_t$ ut fra {i} og {e} rammen hvor et fly har tiden t_0. {t} rammen beskriver referansen til BODY-aksekorset i forhold til en flybane og har

$$\underline{g}^t = \begin{bmatrix} 0 \\ 0 \\ -9{,}81\, m/s^2 \end{bmatrix} \tag{2.29}$$

En applikasjon for android smarttelefoner, kalt "*GPS Status*" gjør det mulig å lokalisere UNIK's nøyaktige posisjon på jordoverflaten. Med andre ord, plassering av tangentrammen.



Figur 2.2.3: GPS lokalisering

Programvaren fanger opp 5 av 8 satellitter som brukes til referansepunkter, og gir breddegrad (latitude) på 59,97598˚N og lengdegrad (longitude) på 11,04529˚Ø. Ut fra disse verdiene kan det konkluderes med at tangentrammen og UNIK's lokale posisjon på jordoverflaten ligger 60˚ nordøst for ekvator.

Fartøyet sett i fra tangentrammen er også definert med kun bevegelse i $\vec{x}_t\vec{y}_t$-planet, og en fast høyde. Flyet har ingen dynamikk i det 2-dimensjonale planet og er kun satt til å rotere rundt $\vec{z}_t$-aksen. Avstanden mellom O_t til O_e er gitt ved posisjonen $\underline{p}_t^e$:

$$\underline{p}_t^e = [r_e \cos L\,; 0;\ r_e \sin L] \tag{2.30}$$

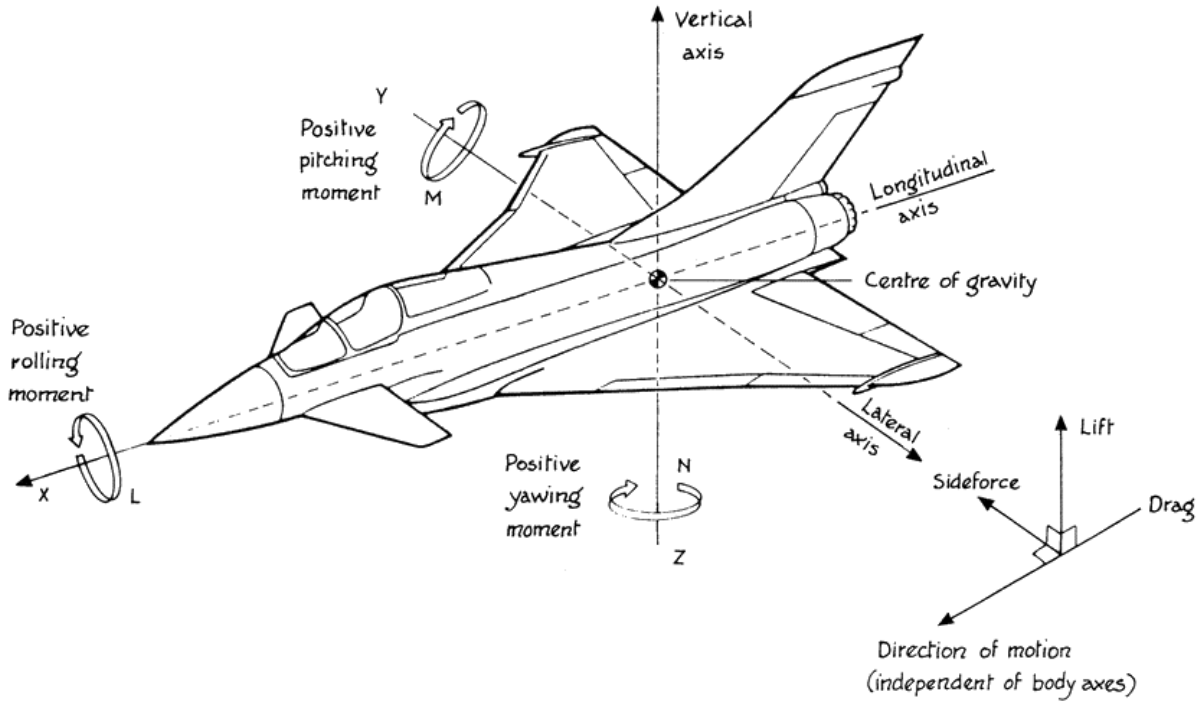
Hvor

$$L = \frac{\pi}{3} = 60^\circ \tag{2.31}$$

2.2.4 Legemeramme – BODY

Koordinatsystemet BODY (Body-fixed)[17] er et fartøyfast koordinatsystem og forflytter seg og roterer med flyets bevegelse. BODY har notasjon {b}, hvor vinkelhastigheter (ω) blir definert i forhold til b-rammen. Denne rammen består av $\vec{x}_b$-, $\vec{y}_b$- og $\vec{z}_b$-akse, hvor $\vec{x}_b$ peker fremover i fartøyet, $\vec{y}_b$ ligger på tvers av legemet med positiv retning langs venstre vinge og $\vec{z}_b$ står normalt oppover fra $\vec{x}_b\vec{y}_b$-planet med origo O^b i massesenteret. Oppsummert beskrives rammen som:

$$F^{\{b\}} = \{O_b, \vec{x}_b, \vec{y}_b, \vec{z}_b\}$$



Figur 2.2.4: BODY-ramme

Fra denne rammen får gyro og akselerometret sine sensitive verdier. Eulervinklene *rull*, *stamp* og *kurs* kan utrykkes ved fartøyets endring med hensyn på tiden.

En oppsummering av konstantene fra de ulike rammene er gitt i tabellen:

Konstant	Beskrivelse
r_e	$6{,}370e^6$ meter
$\underline{p_t^e}$	$[r_e \cos L\,; 0;\ r_e \sin L]$
ω_e	$7{,}292115e^{-5}$ rad/sek
$\underline{\omega_e^i}$	$[0; 0;\ \omega_e]$
g	$9{,}81\ m/s^2$
$\underline{g^t}$	$[0; 0;\ -g]$

Tabell 2.2.1: Oppsummering av konstanter for de ulike rammene

I mange tilfeller ønsker man å referere ulike variabler til de forskjellige rammene som er satt opp for et system, spesielt for et fly som beveger seg med en bevisst hastighet i lufta. Typiske tilstandsvektorer som blir brukt til dette er posisjoner, hastigheter, akselerasjoner, vinkelakselerasjoner, krefter som påvirker legemet, rotasjoner og mye mer. En ramme kan kobles opp mot neste ramme som ligger nærmest, men også refereres til rammen som ligger lengst unna. Denne sammenhengen benyttes mye ved direkte oppkobling mellom to bestemte rammer eller flere, hvor informasjonen mellom disse er interessant.

2.2.5 Transformasjon mellom de ulike rammene

For å beskrive fartøyets bevegelse i 6 frihetsgrader er det nødvendig å relatere for eksempel hastigheten og vinkelhastigheten til en referanseramme[18]. Dette gjøres ved å skrive:

v_b^{ac} hastighet av *b* relativt i *a*, dekomponert i *c*

ω_b^{ac} vinkelhastighet av *b* relativt i *a*, dekomponert i *c*

Med andre ord hastigheten eller vinkelhastigheten av punktet *b* i forhold til (relativt) punktet *a* (origo i referanserammen *a*) er *dekomponert* i referanserammen *c*. Dekomponering av vektoren i *c* betyr at man leser av vektorens koordinater lang x-, y-, og z-aksene i det kartesiske koordinatsystemet *c*. Transformering av hastighet og vinkelhastighetsvektorene i ulike rammer kan raskt beskrives ved hjelp av rotasjonsmatriser. To eksempler på dette er:

$$v_b^{ac} = R_b^c v_b^{ab} \tag{2.32}$$

$$\omega_b^{ac} = R_b^c \omega_b^{ab} \tag{2.33}$$

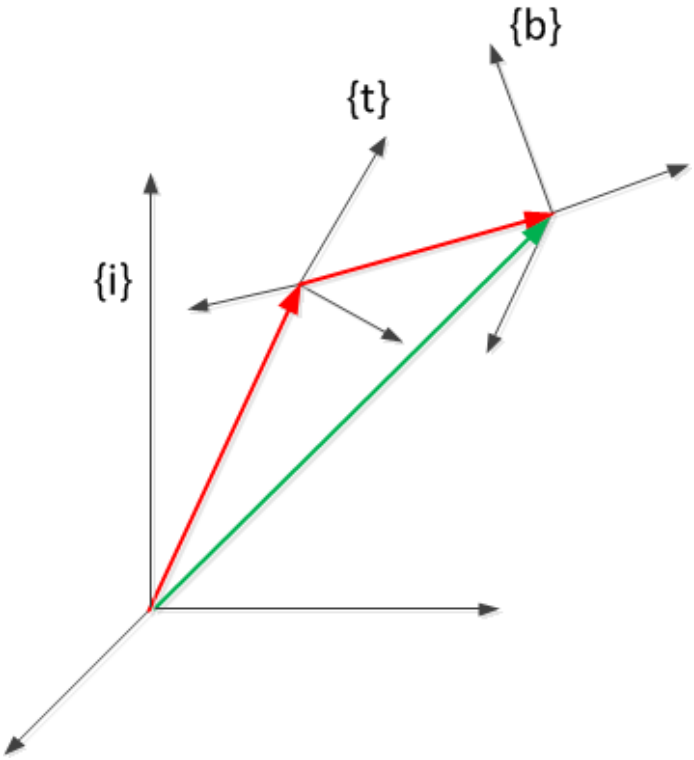
Hvor $R_b^c = R_b^c(\theta_{cb})$ er en 3×3 sammensatt rotasjonsmatrise (RKM) som beskriver sammenhengen mellom en vektor i referansesystemene *b* og *c*. For referansesystemene *b* og *c* er RKM en funksjon av elementene i vektoren $\theta_{cb} = [\varphi, \theta, \psi]^T$, dvs Eulervinklene.

Notasjon ved bruk av mange like subskript innenfor samme ramme, gjør at man kan forenkle tilstandsvektoren til:

$$v_b^{aaa} = v_b^a \tag{2.34}$$

$$\omega_b^{aaa} = \omega_b^a \tag{2.35}$$

Et eksempel på hvordan en vektor kan beskrives fra to rammer til en, er illustrert under i figur 2.2.5



Figur 2.2.5: Sammenhengen mellom de ulike rammene: {i}→{t}→{b} og {i}→{b}

3 Banegenerator

Det er ønskelig å lage en banegenerator i arbeidsverktøyet MATLAB hvor hensikten er å motta verdier som kan testes opp mot en matematisk modell for et TNS.

Interessante verdier som er nyttig å få ut av en slik banegenerator er flyets posisjon(p_b^t), hastighet($\underline{v}_b^t$) og akselerasjon(a_b^t), samt flyets orientering ved bruk av Eulervinkler, vinkelhastighet (ω_b^t) og vinkelakselerasjon i forhold til tiden. I tillegg til dette, vil disse parameterne bli brukt til å regne ut den spesifikke kraften ($\underline{f}^b$) og vinkelhastigheten ($\underline{\omega}_b^{ib}$) til flyet når legemet kjører gjennom banen. Dette blir nærmere beskrevet teoretisk i avsnitt 3.1.2. Denne generatoren lages for å anskaffe informasjon for et fysisk system til et TNS og videreutvikle navigasjonslikninger med et pådrag til begge systemene.

3.1 Teori og fremgangsmåte

Programmet som ønskes baseres på matematiske grunnlag hvor riktig fremgangsmåte og utregninger har stor betydning. I denne delen fremstilles den teoretiske tankegangen bak systemet med matematisk beskrivelse og delvis pseudo kode. Til slutt introduseres programdelen med beregning av ulike funksjoner og variabler. Banen illustreres med grafiske tegninger fra MATLAB.

3.1.1 Fly bane

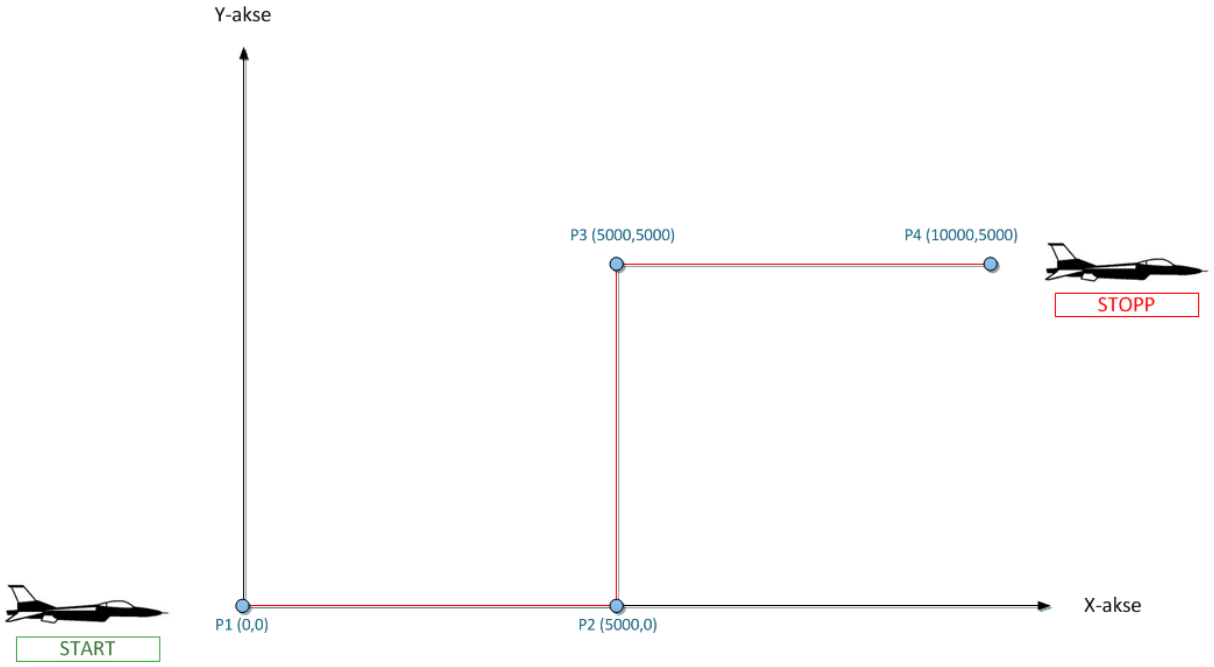
Realisering av banegeneratoren skal starte med at brukeren av programmet velger ut fire valgfrie posisjoner [x,y] som defineres i et 2 dimensjonalt koordinatsystem $x^t y^t$-planet. Punktene som defineres lager rette linjevektorer i planet og navigerer flyet i en bestemt retning. For å illustrere teorien til banen er det tatt hensyn til bestemte koordinater gitt i tabellen. Disse ønskes også å brukes for å beskrive et resultat:

Posisjon	Flyets koordinater (m)
P1	[0,0]
P2	[5000,0]
P3	[5000,5000]
P4	[10000,5000]

Tabell 3.1.1: Banekoordinater

De ulike posisjonene vil danne vektorer som peker ut fra tangentplanet. Høyden fra O_t langs $\vec{z}_t$-aksen opp til flyets posisjon er satt til 1000 meter (1km). For å praktisere dette videre vil banen kun baseres på rette linjer og sirkelbuer, og vil gjøre systemet deterministisk. Følgelig vil banen ikke ha noen form for dynamikk, og løsningen blir av numerisk form.

Grunnen til å benytte sirkler er fordi i praksis vil ikke et fly kunne rotere bestemt fra en rett strekning til en annen uten å kunne svinge jevnt over. Dette vil i praksis oppfylle de fysiske lovene for et fly ved numerisk beregning best mulig, og flyet vil få en mer naturlig overgang. Brukeren definerer flyets konstante hastighet gjennom banen manuelt sammen med andre parametere som blir definert i avsnitt 3.1.2 for en 2-dimensjonal flykontroller. Denne kontrolleren skal illustrere fly gjennom banen.



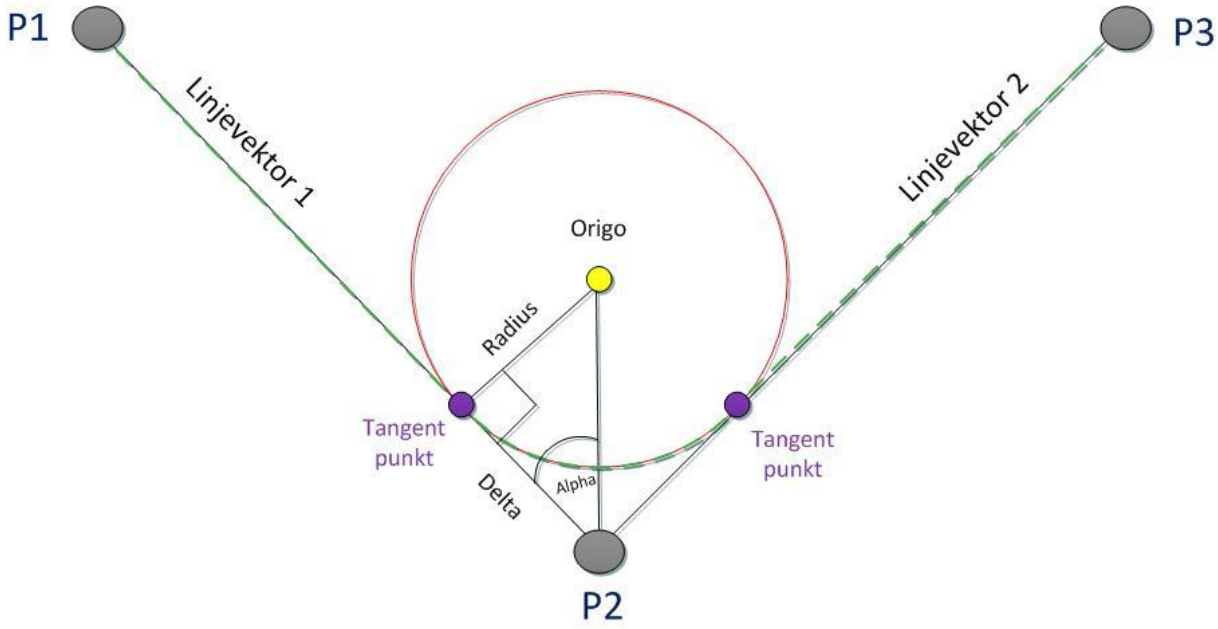
Figur 3.1.1: Bane med gitte koordinater

Ved å legge til sirkelbuer mellom posisjonskoordinatene, begrenses fokuset kun til forholdet mellom tre punkter i banen. Her beregnes vinkelen mellom to vektorer, og en sirkel med konstant radius som danner to tangentpunkter (TP1 og TP2) langs vektorstørrelsene. Dette er en generell sekvens som repeteres gjennom hele banen. For å kunne utføre dette må det først defineres to veipunkter (P1 og P3) som ekspanderer ut fra det samme opprinnelige veipunktet i et aksesystem (P2). Ved å trekke fra det nåværende veipunktet fra de to andre veipunktene, vil dette produsere to linjevektorer:

$$\overrightarrow{LV1} = P_1 - P_2 \tag{3.1}$$

$$\overrightarrow{LV2} = P_3 - P_2 \tag{3.2}$$

Ved å trekke en linje fra det opprinnelige veipunktet P2 til sirkelorigo, vil dette danne vinkelen alpha (α) mellom denne strekningen og linjevektor $\overrightarrow{LV1}$. Dette utgjør halve vinkelen mellom linjevektorene.



Figur 3.1.2: Bue analyse

Alpha beregnes ved følgende likning:

$$\alpha = \frac{1}{2}\cos^{-1}\left(\frac{\overrightarrow{LV1} * \overrightarrow{LV2}^T}{\left\|\overrightarrow{LV1}\right\| * \left\|\overrightarrow{LV2}\right\|}\right) \tag{3.3}$$

Sirkelorigo defineres med vanlig trigonometrisk beregning av en rettvinklet trekant. Katetene til en rettvinklet trekant i dette tilfellet er delta (Δ) og radiusvektor fra sirkelen. Radius er noe som brukeren definerer selv, beskriver hvor stor sirkelen skal være, og hvor stor sirkelbuen blir. Hvis radiusen blir for stor i henhold til banesvingene, kan dette føre til at sirkelbuene overlapper hverandre. Det er viktig at radiusvektoren er like stor som deltavektoren. For å finne delta, brukes de kjente variablene i trigometrisk beregning:

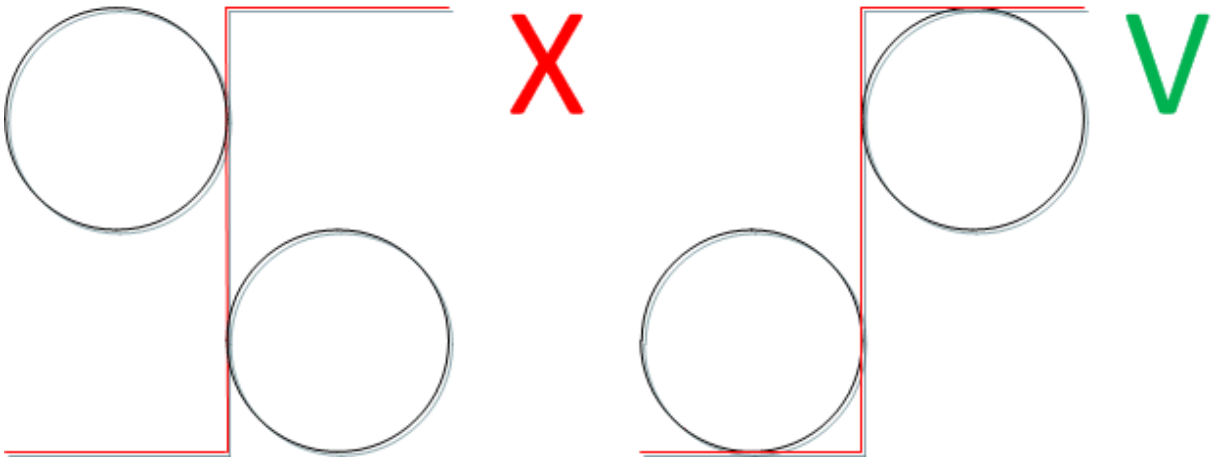
$$\Delta = \tan\left(\frac{\pi}{2} - \alpha\right) * Radius \tag{3.4}$$

Neste steg er å finne sirkelorigo ved å bevege seg langs $\overrightarrow{LV2}$ med en lengde som tilsvarer delta. Med andre ord, til siste tangentpunkt er nådd. Videre flyttes en lengde lik radius i en vinkelrett retning. Er begge tilfellene like, er sirkelorigo definert riktig.

I tillegg er det viktig å se til at sirkelen mellom linjevektorene er riktig plassert. Ønsket er å plassere sirkelen på innsiden, og for å kunne realisere dette må innsiden defineres. Dette gjøres ved å kalkulere kryssproduktet[6] av $\overrightarrow{LV1}$ og $\overrightarrow{LV2}$.

$$Kryssprodukt = \left(\overrightarrow{LV1} \times \overrightarrow{LV2}\right) = \left(\overrightarrow{LV1_x} * \overrightarrow{LV2_y} - \overrightarrow{LV1_y} * \overrightarrow{LV2_x}\right) \tag{3.5}$$

Er kryssproduktet positivt og vektorene sees fra startpunkt av vektor $\overrightarrow{LV1}$ mot $\overrightarrow{LV2}$, peker $\overrightarrow{LV2}$ på venstre side av $\overrightarrow{LV1}$, og motsatt når kryssproduktet er negativt. Hvis kryssproduktet er lik null, ansees $\overrightarrow{LV1}$ og $\overrightarrow{LV2}$ som en rett linje, eller parallelle. Dette betyr at fortegnet til kryssproduktet bestemmer om sirkelenheten er på riktig side av begge vektorene.



Figur 3.1.3: Kryssprodukt

Negativt kryssprodukt betyr også at den ortogonale vektoren er på høyre side av $\overrightarrow{LV1}$, og på venstre side hvis kryssproduktet er positivt. Den ortogonale vektoren er lik enhetsvektoren[4] av $\overrightarrow{LV1}$ hvor x og y skifter plass, og fortegnet forandres på enten x- eller y-komponenten.

$$\overrightarrow{LV1} = [x, y] = [y, x] \tag{3.6}$$

[4] En enhetsvektor er en vektor med lengde lik 1. En operasjon der en vilkårlig egentlig vektor deles med sin egen lengde kalles å normaliserevektoren, og resultatet av operasjonen er en enhetsvektor[19].

Enhetsvektoren regnes ut på dette matematiske grunnlaget:

$$e = \frac{\overrightarrow{LV1}}{\left\|\overrightarrow{LV1}\right\|} \tag{3.7}$$

Dette er avhengig av hvilken side sirkelorigo er definert. Dette er et spesialtilfelle av rotasjonsmatrisen hvor rotasjonen er -90˚ eller 90˚ når x og y skiftes, og ved -90˚ gir det en negativ y verdi (-y).

Siste operasjon er å bevege seg direkte fra punkt P2 til sirkelorigo. Dette gjøres ved å beregne:

$$Origo = P2 + (e * \Delta) + (ortonormalvektor * Radius) \tag{3.8}$$

Hver enkel sirkelorigo som beregnes kan bli lokalisert og brukt som referansepunkt til å estimere en sving i flybanen.

Flere gjentakelser av denne teoretiske sekvensen, vil gjøre at flybanen fungerer optimalt med gitte posisjonskoordinater. Flyet vil ha en retningsorientert bane å forholde seg til, med et startpunkt og stoppunkt.

For å finne de ulike tilstandene fra flyets {b}-ramme til {t}-rammen, som ble omtalt innledningsvis, må flyets posisjon og orientering finnes ved hvert tidssample fra start til slutt. Dette gjøres ved hjelp av en 2-dimensjonal flybane kontroller.

3.1.2 2-dimensjonal flybane kontroller

Flybanekontrolleren bygges opp av to type moduler, en for å følge rette linjer og en for å følge svinger. I de ulike modulene er det gitt at forskjellige betingelser må tas hensyn til ettersom hvilken retning flyet kjører. Beveger legemet seg langs en rett strekning eller langs en sving? I modulene ønskes det at posisjonen registreres med tanke på tiden og gir en tilbakemelding når flyet endrer retningstilstand, slik at flyet skifter modul til å bevege seg rett frem eller starte i en sving. Registrering av posisjon langs banen gir flyet en verdi å sammenlikne referansepunktene med og for å vite når en ny modul skal inntreffe. Referansepunktene for flyet blir tangentpunktene. Hver modul har også hver sin rolle når det gjelder fysiske lover for rette strekninger og svinger.

For rette strekninger beregnes posisjon, strekning, hastighet og akselerasjon med tanke på likningene (3.9), (3.10), (3.11) og (3,12) [20]:

Posisjon:

$$x = x(t) \tag{3.9}$$

Strekning/forflytning:

$$\Delta x = x - x_0 \tag{3.10}$$

Hastighet:

$$v = \frac{dx}{dt} \tag{3.11}$$

Akselerasjon:

$$a = \frac{dv}{dt} = \frac{d^2x}{dt^2} \tag{3.12}$$

Samtidig tar strekningen hensyn til rotasjonen i banen. Siden banen konsentreres kun om $\vec{x}_t\ \vec{y}_t$ -planet brukes en 2-dimensjonal rotasjonsmatrise på formen:

$$R = \begin{bmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{bmatrix} \tag{3.13}$$

Med Eulervinkler vil flyet kun rotere om $\vec{z}_t$-aksen henvist til likning (2.21) med hensyn på tiden. Likning (2.22) og (2.23) vil ikke ha noen vinkelforskjell. Siden flyet er satt til kun å rotere rundt $\vec{z}_t$-aksen, vil ikke dette være særlig realistisk da banen ikke har noen form for dynamikk.

For svinger beregnes også posisjon, strekning, hastighet og akselerasjon, men i dette tilfellet endres de fysiske lovene fra rette linjer til strekning i sirkler. Tilstandene posisjon, strekning og hastighet regnes ut som tidligere, men akselerasjonen endrer stilling til sentripetalakselerasjon[5].

[5] Sentripetalakselerasjon er akselerasjon inn mot sentrum av en krum bane. Sentripetalakselerasjon oppstår når summen av alle kreftene på et legeme står vinkelrett på retningen, med andre ord at akselerasjonsvektoren står normalt på fartsvektoren[21].

$$a = \frac{v^2}{r} \tag{3.14}$$

Når kontrolleren har kjørt gjennom gitte likninger og betingelser, vil p_b^t, v_b^t og a_b^t være registrert sammen med vinkelposisjon, vinkelhastighet ω_b^t og vinkelakselerasjon. Ved å 1. og 2. derivere vinkelen for posisjon med hensyn på tiden, vil vinkelen for hastighet og akselerasjon fra {b} til {t} rammen, bli definert[22].

Vinkelendring:

$$\Delta\theta = \theta - \theta_0 \tag{3.15}$$

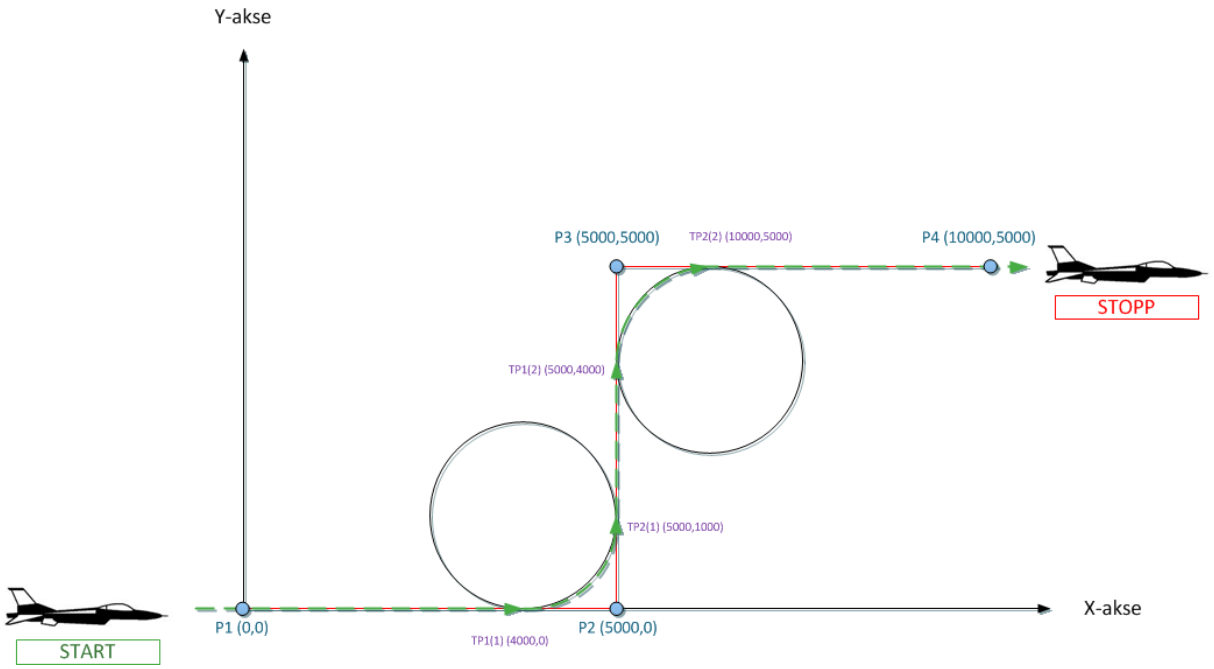
Vinkelfart:

$$\omega = \omega_b^t = \frac{d\theta}{dt} \tag{3.16}$$

Vinkelakselerasjon:

$$\omega = \frac{d^2\theta}{dt^2} \tag{3.17}$$

Med riktig fremgangsmåte og teoretisk bakgrunn, skal flyet kunne posisjoneres gjennom en estimert flybane med posisjonskoordinater som retningslinje. Figur 3.1.4 viser visuelt tanken for banegeneratoren.



Figur 3.1.4: Ønsket fly bane

Kriteriet for generatoren er at flyet skal gjennomføre banen på ca. 60 sekunder fra start til stopp med en samplingstid på 0,1 sekunder. Med denne informasjonen og det matematiske grunnlaget som er definert, er det mulig å finne strekning og hastighet som må initialiseres før kontrolleren startes:

$$Strekning = \left(\left(\frac{2\pi r}{4}\right) * x_{svinger}\right) + \sum|\overrightarrow{LV}| \tag{3.18}$$

$$Hastighet = \frac{Strekning}{Tid} \tag{3.19}$$

Initialisering for kontrolleren blir som følger:

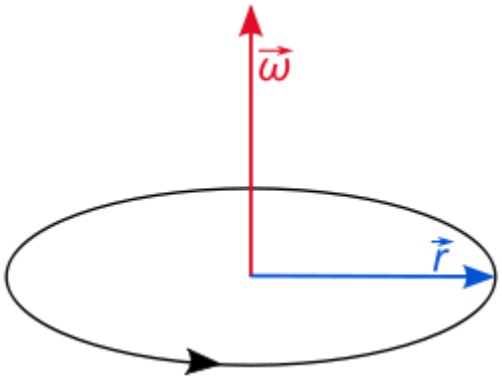
Verdi	**Beskrivelse**
0,1	Samplingstid
60 s	Tid, fra start til stopp i sekunder
14140 m	Total lengde på fly banen
235 m/s	Hastigheten til flyet

Tabell 3.1.2: Initialiseringsverdier for fly kontrolleren

3.1.3 Flyets spesifikke kraft og vinkelhastighet

På bakgrunn av flybanen, estimeres en spesifikk kraft ($\underline{f}^b$) og en vinkelhastighet ($\underline{\omega}_b^{ib}$) representert i {b} aksesystemet sett fra {t} rammen.

Generelt i fysikken er vinkelfart[23] ω en vektor mengde. Denne beskriver hastigheten på rotasjonen og orienteringen til aksesystemene i de forskjellige rammene hvor objekt befinner seg. SI enheten for vinkelfarten er radianer per sekund (rad/s) og retningen på vinkelhastighetsvektoren (pseudovektor) vil være langs rotasjonsaksen.



Figur 3.1.5: Vinkelhastighetsvektoren

Fra tidligere er det bare z-aksen som roterer i jordrammen i forhold til {t} rammen, og vinkelhastigheten for dette tilfellet kan dermed beskrives:

$$\underline{\omega}_e^i = \begin{bmatrix} \underline{\omega}_{e,X}^i \\ \underline{\omega}_{e,Y}^i \\ \underline{\omega}_{e,Z}^i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 7{,}292115e^{-5} \end{bmatrix} \tag{3.20}$$

Vinkelforholdet mellom {t} rammen og jordrammen {e} har ingen endring, siden tangentplanet alltid forflytter seg sammen med jordrotasjonen og kan dermed skrives

$$\underline{\omega}_t^e = \begin{bmatrix} \underline{\omega}_{t,X}^e \\ \underline{\omega}_{t,Y}^e \\ \underline{\omega}_{t,Z}^e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.21}$$

Vinkelhastigheten $\underline{\omega}_b^t$ blir beregnet fra banekontrolleren og representerer hastighetsrotasjonen i {b}, relativt i {t}. For å beregne $\underline{\omega}_b^{ib}$ må vinkelhastighetene som er definert, dekomponeres og transformere fra {b} rammen til {i} rammen. Dette gjøres ved rotasjonsmatriser.

Rotasjonsmatrisen fra {i} til {e}, med rotasjon rundt z-aksen til {e}, gjør at R_e^i kan utrykkes som følgende matrise:

$$R_e^i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\underline{\omega_e} * t) & -s(\underline{\omega_e} * t) \\ 0 & s(\underline{\omega_e} * t) & c(\underline{\omega_e} * t) \end{bmatrix} \tag{3.22}$$

Ramme {i} og {e} faller sammen ved t_0 og endres når tiden øker sammen med $\underline{\omega_e}$. Den teoretiske og matematiske bakgrunnen for RKM og jordrammen bekrefter denne sammenhengen.

Rotasjonsmatrisen for ramme {e} til {t} beskrives med en rotasjon rundt $\vec{y}_t$-aksen. Dette er vinklingen tangentrammen har relativt til jordrammen og beskriver vår lokasjon på jordoverflaten.

$$R_t^e = R(\theta_3, z)R(\theta_3, x - L) = \begin{bmatrix} 0 & -s(L) & c(L) \\ 1 & 0 & 0 \\ 0 & c(L) & s(L) \end{bmatrix} \tag{3.23}$$

Hvor L er gitt ved definisjonen (2.31)

Rotasjonsmatrisen fra {b} til {t} er bestemt av Eulervinklene. Siden det er bestemt at flyet ikke skal rotere langs $\vec{y}_b$ vil rotasjon langs $\vec{y}_b$-aksen $\theta_2 = \theta$ nullstilles. Det samme gjelder for rotasjonen rundt $\vec{x}_b$-aksen $\theta_1 = \varphi$. Med andre ord, siden denne oppgaven ikke har noe dynamikk, vil ikke flyet ha noe rull- og stamprotasjon. Matrisen regnes dermed kun om $\vec{z}_b$ med kurs rotasjon $\theta_3 = \psi$ som kommer fra banegeneratoren. Dette er flyets posisjon beskrevet med vinkler gjennom banen.

$$R_b^t = R(\psi)R(\theta)R(\varphi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\varphi - s\psi c\varphi & c\psi s\theta c\varphi + s\psi s\varphi \\ s\psi c\theta & s\psi s\theta s\varphi + c\psi c\varphi & s\psi s\theta c\varphi - c\psi s\varphi \\ -s\theta & c\theta s\varphi & c\theta c\varphi \end{bmatrix} \tag{3.24}$$

Hvor R_b^t kan transformeres, og vil da få

$$R_t^b = (R_b^t)^T \tag{3.25}$$

Hittil kan R_b^e regnes ut med tanke på R_t^e og R_b^t som lager en rotasjonsmatrise som beveger seg direkte fra {e} rammen til {b} rammen.

$$R_b^e = R_t^e R_b^t \tag{3.26}$$

Transformasjon av R_b^e gir:

$$R_e^b = (R_b^e)^T \tag{3.27}$$

Nå er de viktigste rotasjonsmatrisene definert og som brukes for å representere flyet i de ulike rammene. For å kombinere den direkte rotasjonen fra {b} dekomponert i {i} rammen, multipliseres rotasjonsmatrisene fra hver ramme sammen:

$$R_b^i = R_e^i R_t^e R_b^t \tag{3.28}$$

Hvor R_b^i også kan transformeres og vil gi:

$$R_i^b = (R_b^i)^T \tag{3.29}$$

De ulike vinkelhastighetene kan skrives på følgende måte ved å dekomponere og transformere vektorene over i {b} rammen. Dette gjøres ved bruk av RKM:

$$\underline{\omega}_e^{ib} = R_i^b \omega_e^i \tag{3.30}$$

$$\underline{\omega}_t^{eb} = R_e^b \underline{\omega}_t^e \tag{3.31}$$

$$\underline{\omega}_b^{tb} = R_t^b \underline{\omega}_b^t \tag{3.32}$$

Ved å addere sammen vinkelhastighetene gitt over, vil den sanne vinkelhastigheten $\underline{\omega}_b^{ib}$ fra {b} til {i} rammen bli funnet.

$$\underline{\omega}_b^{ib} = R_t^b \underline{\omega}_b^t + R_e^b \underline{\omega}_t^e + R_i^b \omega_e^i \tag{3.33}$$

Eller ved:

$$\underline{\omega}_b^{ib} = \underline{\omega}_b^{tb} + \underline{\omega}_t^{eb} + \underline{\omega}_e^{ib} \tag{3.34}$$

Denne tilstanden spiller en viktig rolle for TNS i det fysiske- og mekaniserte systemet, sammen med den spesifikke kraften. Disse verdiene kan også hentes ut fra gyroer. Gyroer måler vinkelhastighet relativt til treghetsrommet.

Spesifikk kraft[24], også kalt g-krefter og masse-spesifisert kraft, er målt (m/s^2) som også er enheten for akselerasjon. Bokstavelig talt er ikke spesifikk kraft egentlig en type kraft, men en akselerasjon. Den kan representeres i de ulike referanserammene, men består akselerasjonen av koordinater er den rammebetinget. Tilstanden $\underline{f}^b = \underline{f}_b^{ib}$ og beskriver den spesifikke kraften av {b}-aksesystemet, relativt i jordramma {i}, dekomponert i {b} ramma. Spesifikk kraft er også noe som kan kobles opp mot et akselerometer, som blir bedre beskrevet lengre ut i oppgaven. Kort forklart gir denne enheten målinger på objektets akselerasjon og gravitasjonskraft (Einsteins ekvivalensprinsipp[6]).

Akselerasjon for dette rammesystemet blir gitt av $\underline{a}_b^i$, og må i tillegg roteres med en rotasjonsmatrise R_i^b som definert ved likning (3.29). Akselerasjonen beregnes ut fra likningsoppsettet for teorem A.19: *Hastighet og akselerasjon sett fra flere koordinatsystemer med relativbevegelse (algebraiske vektorer)*[25]

$$\underline{p}_b^i = R_e^i p_t^e + R_e^i R_t^e \underline{p}_b^t \tag{3.35}$$

$$\underline{v}_b^i = S\left(\underline{\omega}_e^i\right) R_e^i \underline{p}_t^e + S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{p}_b^t + R_e^i R_t^e \underline{v}_b^t \tag{3.36}$$

$$\underline{a}_b^i = S\left(\underline{\omega}_e^i\right)\left\{S\left(\underline{\omega}_e^i\right) R_e^i \left[\underline{p}_t^e + R_t^e \underline{p}_b^t\right] + 2R_e^i R_t^e \underline{v}_b^t\right\} + R_e^i R_t^e \underline{a}_b^t \tag{3.37}$$

Hvor $S\left(\underline{\omega}_e^i\right)$ beregnes matematisk:

$$S\left(\underline{\omega}_e^i\right) = \begin{bmatrix} 0 & -\underline{\omega}_{e,Z}^i & \underline{\omega}_{e,Y}^i \\ \underline{\omega}_{e,Z}^i & 0 & -\underline{\omega}_{e,X}^i \\ -\underline{\omega}_{e,Y}^i & \underline{\omega}_{e,X}^i & 0 \end{bmatrix} \tag{3.38}$$

I tillegg til akselerasjonen spiller gravitasjonskraften en viktig rolle for den spesifikke kraften. Denne gravitasjonskraften er definert i {t} rammen, og for å få denne vektoren over i {b} rammen, må $\underline{g}^t$ roteres med rotasjonsmatrisen R_t^b. Denne vektoren har en negativ akselerasjonskraft på $-9{,}81\ m/s^2$, akkurat samme tilstand den har i {t} rammen, og er gitt ved:

[6] Ekvivalensprinsippet sier at en person innelukket i en boks uten mulighet til å se ut ikke vil kunne skille mellom om boksen står på bakken her på jorda eller om den akselererer ute i verdensrommet med 9,81 m/s2. Einstein konkluderte med at scenarioene i prinsipp er ekvivalente, og at gravitasjon er en fiktiv kraft[26].

$$\underline{g}^b = R_t^b \underline{g}^t = \begin{bmatrix} 0 \\ 0 \\ -9{,}81 \end{bmatrix} \tag{3.39}$$

Den spesifikke kraften $\underline{f}^b$, som brukes videre til et TNS sammen med vinkelhastigheten $\underline{\omega}_b^{ib}$, kan dermed utrykkes med de ferdig utledede likningene (3.29), (3.37), (3.25) og (3.39), og får da likning (3.40):

$$\underline{f}^b = R_i^b \underline{a}_b^i + R_t^b \underline{g}^b \tag{3.40}$$

Den teoretiske delen for ønsket generator og påvirkningskrefter er ferdig beskrevet, og sammen med matematiske utledninger vil dette vise fremgangsmåten som vil bli benyttet for gjennomføre dette. Likningene blir testet opp mot et matematisk brukerverktøy, og vil kunne gi tilstandene som er forklart en brukbar og fornuftige verdi i form av skalaer eller matriser. Hensikten blir å plassere ønskede verdier i et TNS, og sette opp navigasjonslikninger for det fysiske og mekaniserte systemet ved bruk av navigasjonslikninger.

Hittil har teorien og fremgangsmåten basert seg på å lage en komplett flybane med sanne verdier. Neste steg blir å teste og se om teorien kan muliggjøre ønsket som er fremstilt, både for banegeneratoren, flyets spesifikke kraft og vinkelhastighet.

3.2 Beskrivelse av matlab program

MATLAB programmet som er laget er bygget opp av en hoved M-fil som består av et antall funksjoner. Funksjonene består av bestemte algoritmer som er basert på teorien omtalt i del 3.1 og er delt inn ulike kategorier for å gi en oversikt over hvilke tilstander som må beregnes først. Hovedfilen viser hvordan de ulike funksjonene henger sammen og hvilken sekvens som må starte. Filen heter:

- `Main.m`

Med initialverdier oppgitt i tabell 3.1.1 og 3.1.2 sendes disse til første funksjon, ved et kall på funksjonen, og returnerer verdier som trengs for å foreta nye beregner. Viktige variabler som er interessante og må beregnes til et TNS er:

- Posisjon: $\underline{\rho_b^t}$
- Hastighet: $\underline{v_b^t}$
- Akselerasjon: $\underline{a_b^t}$
- Vinkelposisjon
- Vinkelhastighet: $\underline{\omega_b^t}$
- Vinkelakselerasjon
- Spesifikk kraft: $\underline{f}^b$
- Vinkelhastighet: $\underline{\omega_b^{ib}}$

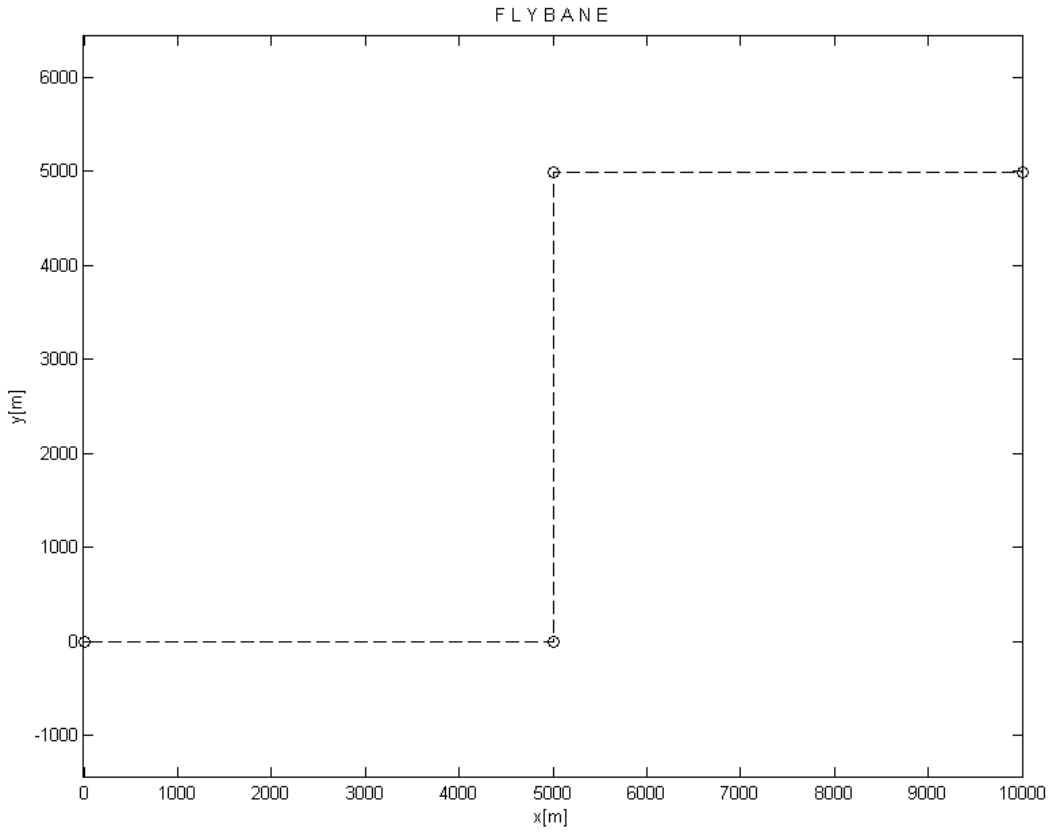
Første funksjonen som kalles er:

- `Banegenerator.m`

Her foretas beregninger som ble fremstilt i den teoretiske delen 3.1.1 og består også av egne funksjoner. Funksjonene i denne generatoren har en vesentlig betydning for å peke flyet i riktig retning, samtidig gi variablene riktig verdi videre. Funksjonene består av

- `Plotsirkel.m`
- `Plotbue.m`
- `Plotlengde.m`

Brukeren av programmet kan selv velge ut hvilke fire koordinater som flyet skal kjøre mot og ha som retningsmål. Dette gjøres manuelt i main-filen ved å endre på posisjonsverdiene i x- og y-retning [x,y]. Koordinatene plasseres i $\vec{x}_t\vec{y}_t$-planet. I dette tilfellet brukes samme koordinater som henvist i teoridelen i tabell 3.1.1.

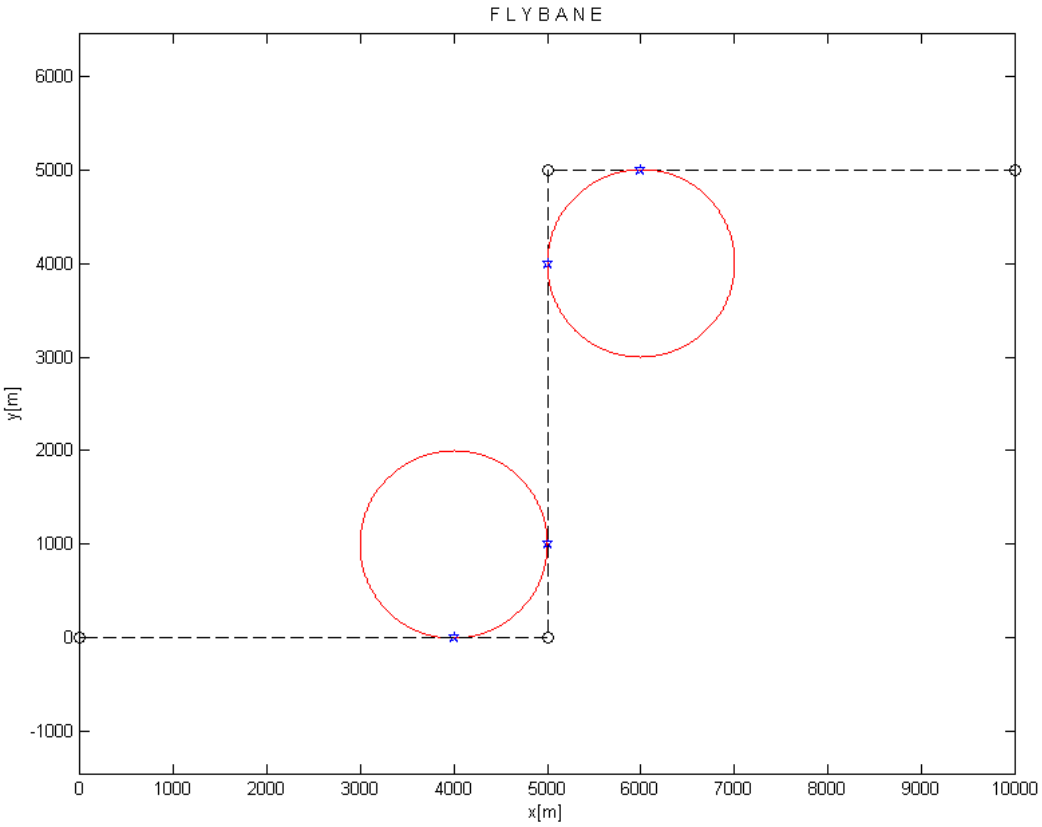


Figur 3.2.1: Retteningsvektorer i tangentplanet med posisjonskoordinater

Fra figur 3.2.1 genereres punktene fra algoritmen og plottes ut som "○" der brukeren har oppgitt koordinatene. Den strippede linjen er til for å vise retningen til flyet, men for å realisere flybanen blir det utført bestemte utregninger og algoritmer bestående av vektorer og matriser. Funksjoner rettet mot dette produserer linjer mellom to punkter og sirkelbuer mellom tre punkter. Mer utfyllende beskrivelse av funksjonaliteten til hver funksjon vil bli beskrevet og fremstilt videre.

3.2.1 Plotsirkel.m

`Plotsirkel.m` funksjonen utfører beregning av en sirkel der tre koordinater til sammen lager en vinkel på 0˚til ±90˚. I tillegg tegnes sirklene opp for å indikere hvor tangentpunktene skjærer banen. Tangentpunktene blir beregnet i `Banegenerator.m` og lagret i hver sin matrise `TP1` og `TP2`.



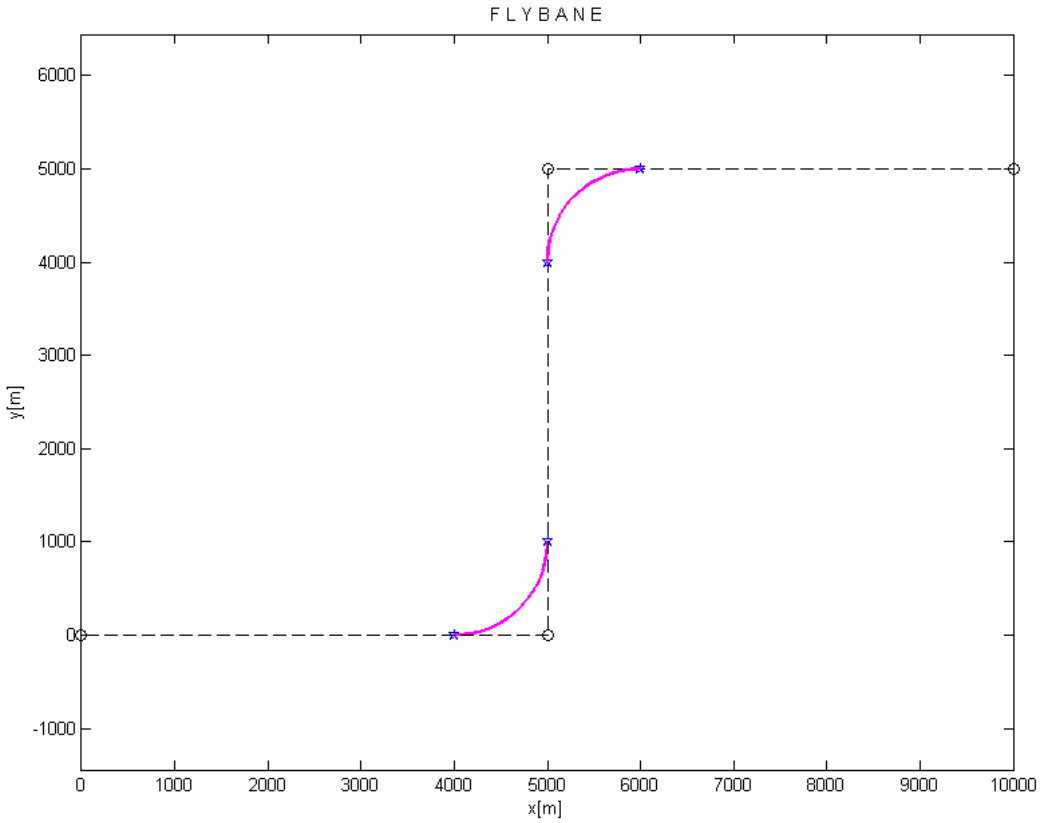
Figur 3.2.2: Sirkelplot.m

Beregning av kryssproduktet og sammenlikning med ortogonalvektorene, fremstiller sirklene på riktig plassering som vist i figuren 3.2.2. I tillegg til utregningene av `alpha` og `delta` defineres sirklenes sentrum, `Origo`, og brukes videre til andre funksjoner.

3.2.2 Plotbue.m

`Plotbue.m` tegner opp sirkelbuer som visualiserer svingene i banen. Ved bruk av hjelpefunksjonen `atan2` i matlab beregnes hvert enkelt delta steg i buelengden med `Origo` som referansepunkt. En matrise ved navnet `Bue` blir definert og kalt inn i funksjonen sammen med variablene `Origo` og `Radius`. Matrisen `Bue` inneholder vinkelinformasjon til flyet før og etter svingene, oppgitt i radianer. Første bue gjør at flyet roterer fra 0˚ til 90˚ og neste bue fra 90˚ til 0˚. I funksjonen bestemmes det om buen skal plottes CCW[7] eller CW med gitte betingelsesalgoritmer. Videre beregnes bue plottet i $\vec{x}_t\vec{y}_t$-planet. Radiusvektoren vil alltid peke innover mot sirkelorigo ettersom buen plottes.

```
x = Origo(1)+Radius*cos(t);
y = Origo(2)+Radius*cos(t);
```

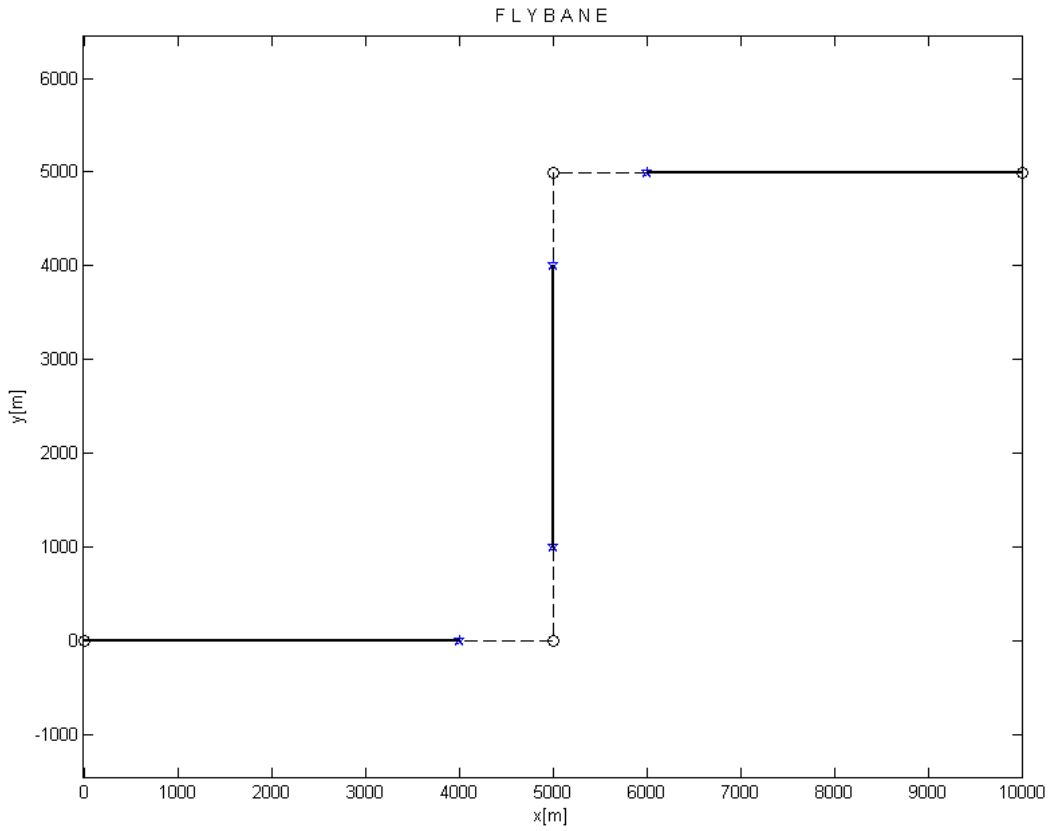


Figur 3.2.3: Bueplot.m

[7] CCW er sirkulær bevegelse i motsatt retning til klokken, og CW er med samme retning som klokken. I en matematisk sammenheng, er en sirkel definert parametrisk i et positivt kartesisk plan med likningene x = sin(t) og y = cos(t), hvor t spores CW ettersom t øker i verdi [27]

3.2.3 Plotlengde.m

Ved bruk av funksjonen `Plotlengde.m` beregnes de rette linjevektorene mellom posisjonskoordinatene og tangentpunktene med matrisene `TP1`, `TP2` og `VB` som initialverdier.

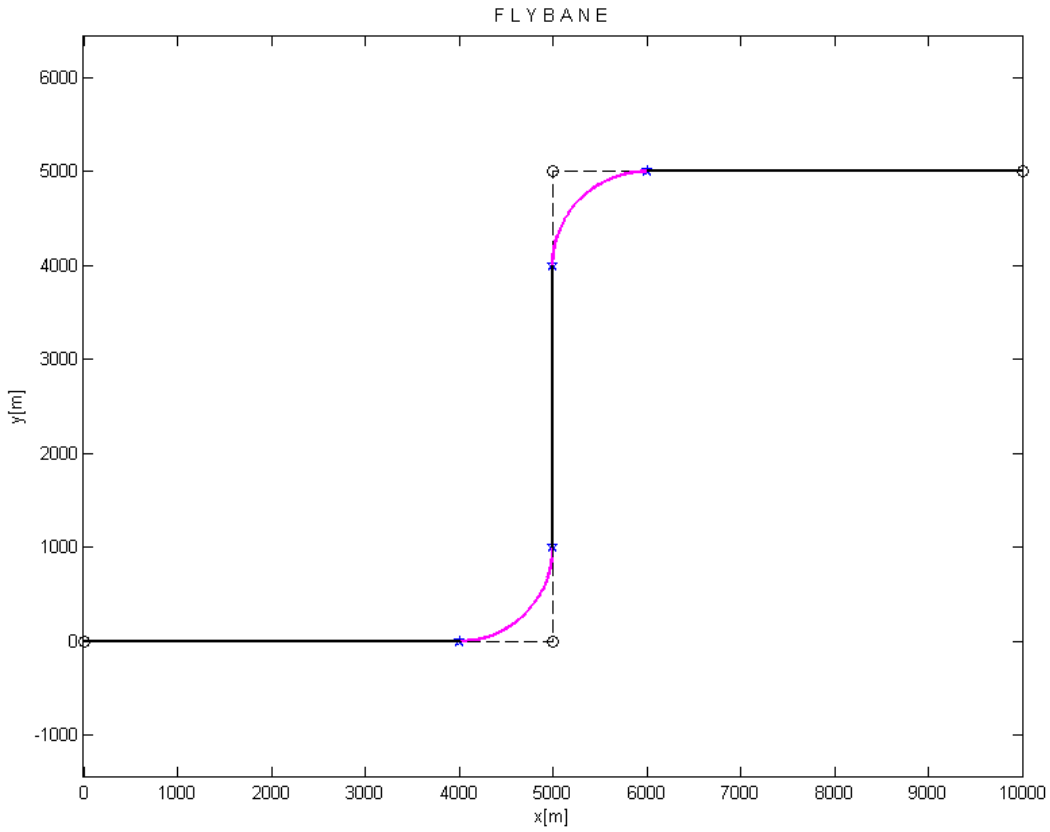


Figur 3.2.4: Plotlengde.m

Linjevektorene `LV` og `LV2` plottes inn i veibanen og kobles opp mot sirkelbuene i tangentpunktene, figur 3.2.4. Vektorene initialiseres i starten av funksjonen før neste betingelse i programmet iverksettes. Neste sekvens som kjøres fra `Banegenerator.m` kalles inn i funksjonen og plotter inn de resterende vektorlinjene.

3.2.4 Banegenerator.m

Etter at alle gitte funksjoner er definert og plassert i riktig rekkefølge er banen klar til å genereres. En *for-løkke* i `Banegenerator.m` utfører et antall iterasjoner som gjør at sekvensene i banen blir beregnet med riktig matriseverdier og kaller på de ulike funksjonene som er definert. Funksjonene fullfører til sammen den komplette banen som ønsket, og bekrefter at sammensetningen av matematiske beregninger stemmer med teorien. Realiseringen av flybanen fremstilles i figur 3.2.5



Figur 3.2.5: Banegenerator.m

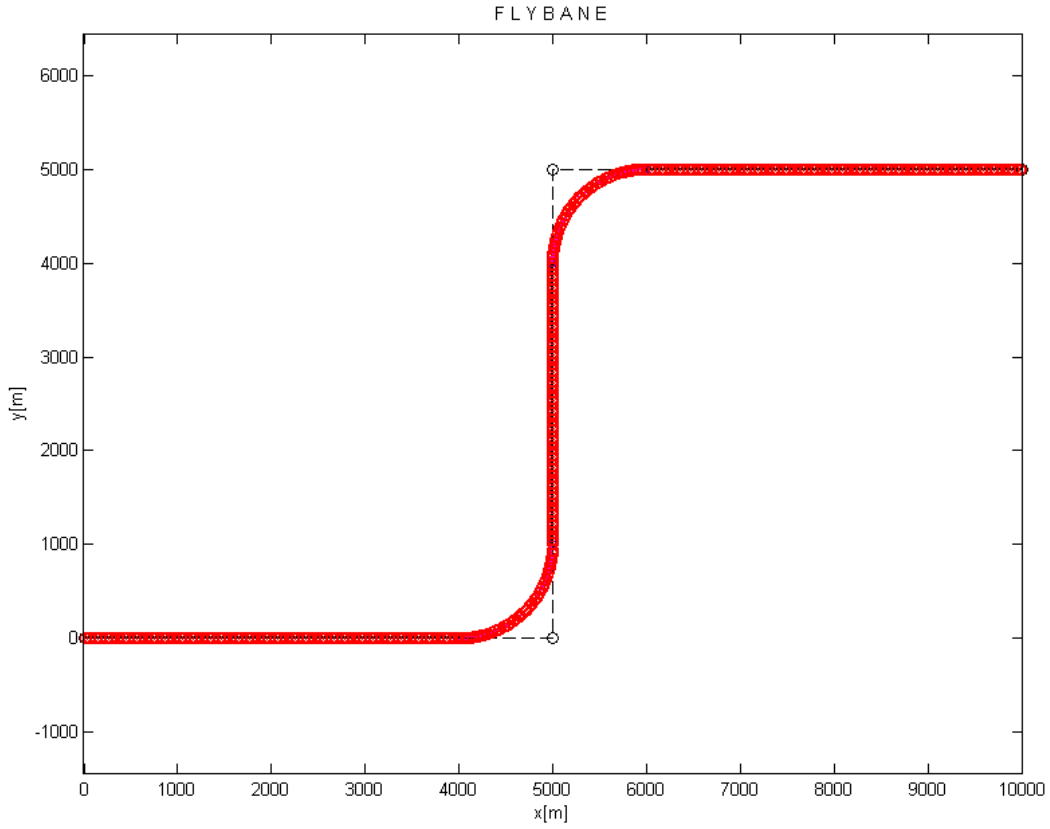
Følgelig lages en 2-dimensjonal kontroller som illustrerer flyets gjennomkjøring i banen sett fra {t} ramma. Dette forklares i funksjonsfila `TrackBane.m`.

3.2.5 TrackBane.m

For å kunne få ut de ønskede tilstandene til flyet er det viktig å registrere flyets orientering og i forhold til {b} og {t} ramma med hensyn på tiden. For å utføre dette, lages det en funksjon som beregner posisjon langs den oppgitte flybanen og registrerer tiden samtidig. Denne funksjonen heter:

- `TrackBane.m`

Bestemte algoritmer sørger for at flyet følger banen med en fast hastighet og samplingstid som blir manuelt satt opp av brukeren. Under baneregistreringen, lagres hvert posisjonsstep i en egen posisjonsmatrise $\underline{p}_b^t$ og tiden lagres i en egen tidsmatrise `Tid`. Denne delen i matlab er delt opp i to moduler som antatt i teoridelen, en registrering for rette linjer og en for svinger. Slike operasjoner gjennomføres ved posisjonskoordinatene og tangentpunktene som referansepunkter.



Figur 3.2.6: TrackBane.m

I første modul analyseres avstanden langs en rett linjevektor til første tangentpunkt. Ved tidspunktet hvor flyet har kommet til ønsket tangentpunkt, skifter flyet modul til bue. Betingelser i bue sekvensen sørger for at flyet kontrollerer seg gjennom svingen og innstilles til å følge en rett linje når neste tangentpunkt er bekreftet. Videre fortsetter flyet i samme gjentatte sekvenser til siste sving er fullført. Et spesialtilfelle inntreffer ved siste linjevektor, her skal flyet stoppe når den har ankommet siste posisjon i banen. Tilstandene som ønskes kan nå trekkes ut og ved hjelp av likningene i avsnitt 3.1.2 for rette strekninger og svinger. Hastigheten er konstant og akselerasjonen varierer ettersom hvilken orientering flyet har. Akselerasjonen er konstant med hastigheten i rette vektorlinjer, men i buer inntreffer igjen sentripetalakselerasjonsprinsippet.

Ved kall av denne funksjonen i main metoden, har programmet følgende variabler i tabellen som kan benyttes videre, hvor N varierer med tiden.

Variabler	**Størrelse**	**Beskrivelse**
`VB`	[4x2]	Bane koordinater {t}
`Radius`	[1x1]	Sving radius {t}
`Origo`	[2x2]	Sirkel origo punkter {t}
`TP1`	[2x2]	Tangentpunt 1 for svingene {t}
`TP2`	[2x2]	Tangentpunt 2 for svingene {t}
`Bue`	[2x2]	Orienteringen til flyet før og etter hver sving {b}
`Tid`	[1xN]	Tiden flyet bruker gjennom banen {t}
`Vinkel`	[1xN]	Flyets vinkelposisjon {b}
`w_b_t`	[1xN]	Flyets vinkelhastighet {b}
`Vinkelakselerasjon`	[1xN]	Flyets vinkelakselerasjon {b}
`p_b_t`	[3xN]	Flyets posisjon
`v_b_t`	[3xN]	Flyets hastighet
`a_b_t`	[3xN]	Flyets akselerasjon

Tabell 3.2.1: Variabel oversikt i Matlab etter kjøring av banegeneratoren

3.2.6 Skjev.m

For å beregne de egne skjevsymmetriske matrisene ble det laget en egen funksjonsfil. Kall på funksjonen i `Simulering.m` gir ut den skjevsymmetriske matrisen av importerte variabler videre til utregning av spesifikk kraft. Vinkelhastigheten $\underline{\omega}_e^i$ er en av tilstandene som omgjøres på skjevsymmetrisk form, funksjonen returnerer $S(\underline{\omega}_e^i)$.

3.2.7 Simulering.m

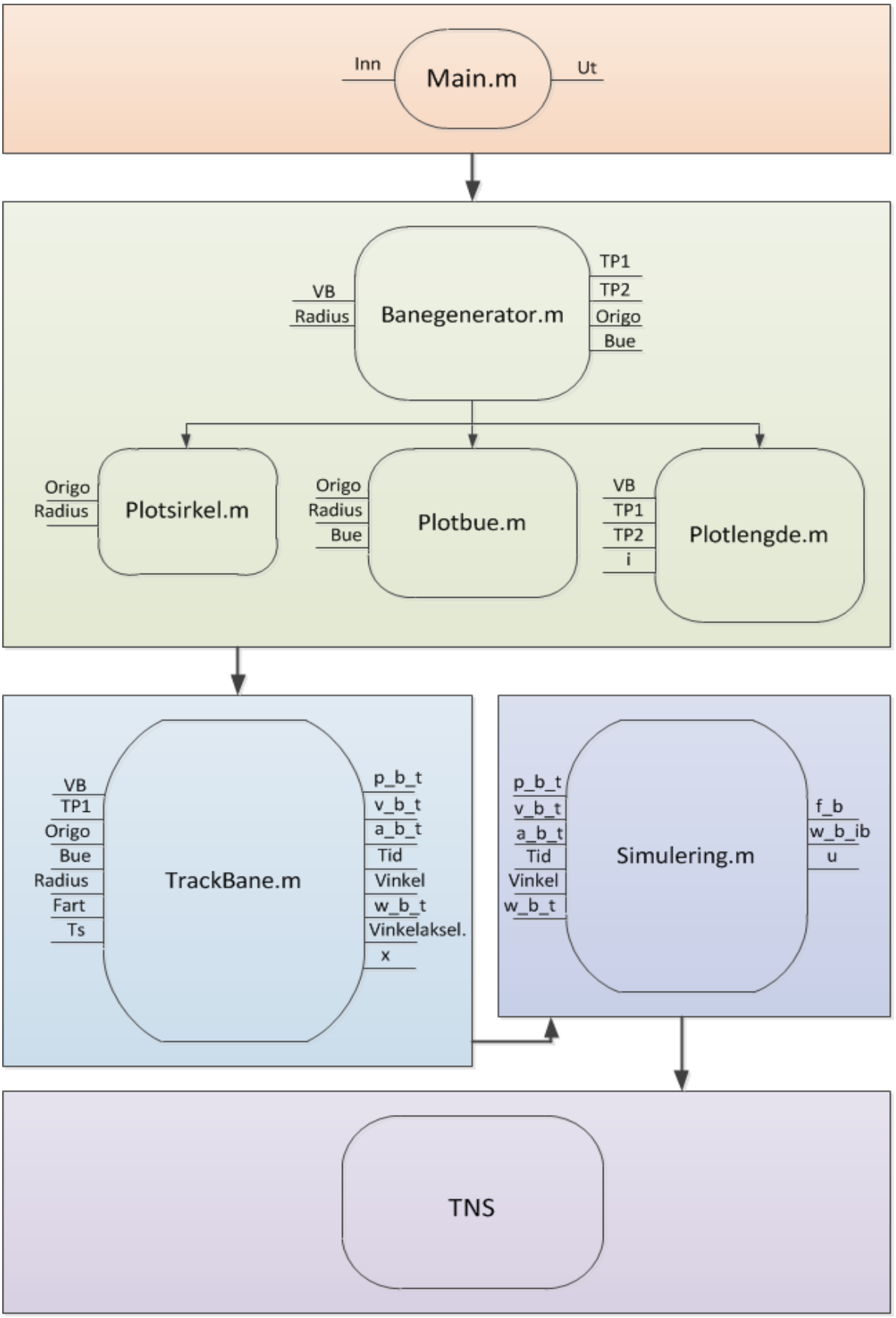
I funksjonsfila `Simulering.m` blir de matematiske likningene for spesifikk kraft og vinkelhastighet beregnet. For at programmet skal utføre utregningene på riktig måte er det viktig at likningene er plassert i riktig rekkefølge. Enkelte variabler trenger en initialiseringsverdi før en lang iterasjon starter, altså en startverdi. Denne iterasjonen gjennomløper variablene som forandrer seg over tid, mens flyet beveger seg gjennom banen. Variablene man får ut av `Simulering.m` er de sanne verdiene `f_b` og `w_b_ib`

Variabler	Beskrivelse
`f_b`	Spesifikk kraft (sann verdi)
`w_b_ib`	Vinkelhastighet (sannverdi)

Tabell 3.2.2: Variabel beskrivelse i MATLAB

3.2.8 Main.m

Nå som alle funksjonene er definert, vil hovedfilen `Main.m` ha en oversikt over alle variablene som er har blitt beregnet. Ved å endre hastighet, svingradius, samplingstid og posisjonskoordinatene i dette feltet, vil hele programmet foreta nye beregninger som er annerledes enn det som tidligere ble estimert. Banegeneratoren vil konstruere banen på en annen måte og svingene vil bli større eller mindre. Når denne delen kjøres, vil hver og en funksjon som er plassert i riktig rekkefølge, utføre beregninger og komme med nye oppdaterte verdier. En oversikt over programmet er illustrert i figure 3.2.7.



Figur 3.2.7: MATLAB program satt sammen av funksjonsblokker

4 Treghetsnavigasjonsystem (TNS)

Treghetsnavigasjon (Inertial Navigation System)[28] er en navigasjonsmetode som bygger på legemers treghet og deres motstand mot endringer i bevegelsestilstanden. Ved hjelp av akselerometre og gyroskoper kan man navigere fartøy (raketter, ubåter og fly) uavhengig av utstyr utenfor fartøyet, for eksempel radiokontakt med en fast stasjon.

Et akselerometer benyttes i denne sammenhengen for å måle akselerasjon, hvor i tillegg egne komponenter til enheten måler gravitasjon. Dette er for å skille akselerasjonene, som skyldes hastighetsforandring og akselerasjonen fra gravitasjonen. Gyroer måler vinkelhastighet og vinkel. Funksjonaliteten og egenskapene til de to enhetene blir forklart nærmere i avsnitt 4.1.1 og 4.1.2 om MEMS.

Sensorenheten IMU (Inertial Measurement Unit)[29],[30] kan bidra til å beskrive et TNS sammen med en programvare som beregner posisjon, hastighet og orientering til et mekanisert system ved hjelp av integrasjonsalgoritmer. IMU består av et kluster, som igjen består av akselerometeret og gyroene som tidligere ble nevnt. Dette klusteret er fastmontert på en felles plattform slik at den relative orienteringen mellom sensorene er lik. TNS kan dermed deles inn to hovedprinsipper:

- Gimbals-basert TNS
- Skrogfast TNS

Gimbals-basert TNS-system kan realiseres ved å montere et akselerometer og en gyro sammen på stabilelementet i en tre- eller fire-akset oppheng som skal gjøre det uavhengig av underlagets dreiebevegelser. Den stabiliserte plattformen (gimbal-systemet) vil da holde konstant orientering i forhold til treghetssystemet på grunn av feilkompenseringen fra gyroen.

Skrogfast TNS benyttes hvis gyro og akselerometeret monteres sammen og festes på fartøyets skrog. De skrogfaste sensorene og treghetssystemet har en sammenheng, og utrykkes ved navigasjonslikningene av differensial form. Disse likningene beskriver bevegelsen til fartøyet i de ulike koordinatsystemene, og det er denne type plattform som er tatt hensyn til i denne oppgaven.

Helt generelt, for mange type teknologiske systemer, er det fordeler og ulemper ved kostbare sensorenheter. Dette gjelder også sensorene brukt for et TNS. Mye av omtalene i denne sammenheng har mye å si for denne teknologien. Meninger og konklusjoner er med på å bidra til at usikkerheter i systemene blir forbedret og at systemene får bedre og robuste løsninger. Dette fører til at flere benytter seg av enhetene, og de blir mer tilpasset brukeren. Fordeler og ulemper ved TNS og sensorene som blir tatt frem i dag er typiske faktorer som[31]:

Fordeler:

- De gir momentan produksjon av legemets posisjon og hastighet.
- De er helt selvstendige.
- Kan benyttes globalt i alle systemer.
- Gir svært nøyaktig asimut og vertikal vektor måling.
- Feilegenskaper er kjent og kan modelleres ganske godt.
- Fungerer godt i hybride systemer.

Ulemper:

- Stilling og hastighetsinformasjonen gir dårlig informasjon ved begrenset tid.
- Utstyret er generelt dyrt og eldre systemer har relativt høye feilrater og dyre å vedlikeholde.
- Nyere systemer er mer pålitelige, men fortsatt dyrt å reparere.
- Akselerometret skiller ikke automatisk eller direkte mellom akselerasjon og gravitasjon, men finner metoder for å løse problemet.

Siden mye av ulempene poengterer at utstyret er dyrt, har hovedfokuset for denne teknologien forflyttet seg til å implementere sensorene over til MEMS-enheter. MEMS-enhetene har lavere produksjonsverdi og er generelt billigere enn andre systemer som baserer seg på et TNS. Dette gir et lavkost TNS som kan løse samme problemstilling i de fleste tilfeller.

Dermed er det interessant å basere TNS på denne type teknologi, hvor man etter hvert kan implementere gyro- og akselerometerdata for å bedømme de ulike sensorene.

4.1 MEMS-sensorer

I denne oppgaven er det tatt hensyn til MEMS gyroskoper og akselerometer ved beregning av et fysisk- og mekanisert system i et TNS.

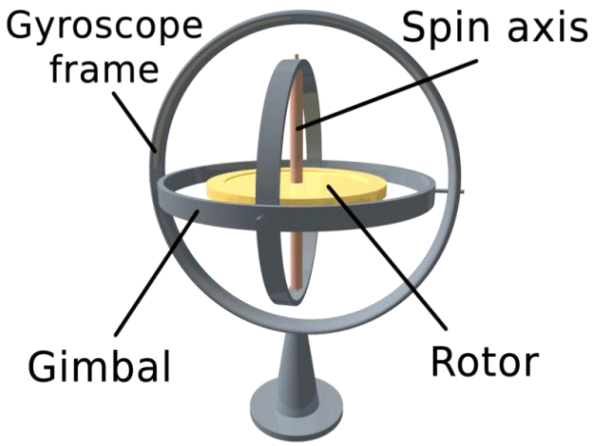
MEMS (Mikro Elektro - Mekaniske Systemer) har fått mye oppmerksomhet de siste tiårene for applikasjoner som krever lavkost og kompakte størrelser med moderate ytelser, som bil, navigasjonssystemer, romforskning, luftfart m.m. Det er også behov for høy-ytelse når det kommer til MEMS for avanserte programmer, og det er en vedvarende verdensomspennende innsats for å utvikle høy-klasse gyroskoper og akselerometre.

Behovet for avansert MEMS krever utfordrende, mekaniske og elektriskdesignede avgrensninger. Optimal ytelse er godt definert, men det er også praktiske problemer knyttet til fabrikasjonsbegrensninger, gevinst/båndbredde betraktninger. I tillegg er det noen ubestemmelige parametere som temperaturavhengighet, drift og forskjøvet vibrasjon, som er vanskelig å regulere bort.

En indikasjon på funksjonaliteten til gyroer og akselerometre, samt hva de består av, beskrives litt nærmere i neste avsnitt. Forholdet disse enhetene har til et TNS er å bruke verdiene til navigasjon, men reelle datamålinger benyttes ikke for denne oppgaven. Dermed kun beregnede datamålinger.

4.1.1 Gyro

Gyro[32], eller gyroskopet er bygget på mange forskjellige måter. Et eksempel på den klassiske varianten til et gyroskop og deres funksjonalitet kan være til stor hjelp for å illustrere dette.



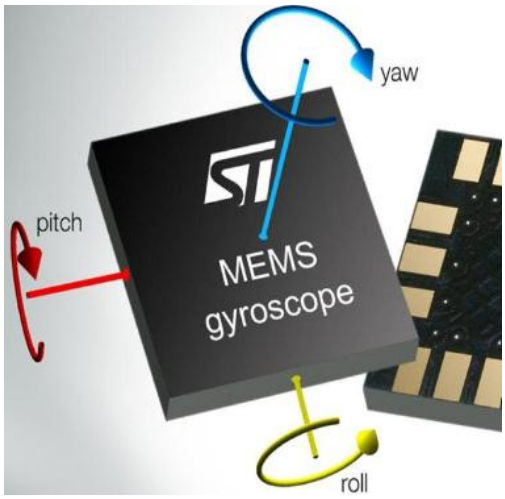
Figur 4.1.1: Gyroskop[33]

Dette gyroskopet er bygget opp av et sentralt hjul eller kalt "rotor" som er montert i en ramme av ringer. Ringene kalles "gimbals" og er enheter som er med på å bygge om strukturen, og tillater hjulet som er sentrert i midten til å bevege seg fritt. Ringene er selv støttet med hver sin spindel eller akse som gjør at rotoren kan bevege seg i 3-dimensjoner. Rammeverket kan dermed skråstilles og rotoren, så lenge den spinner, vil prøve å opprettholde sin posisjon.

Så lenge rotorakselen peker mot sin opprinnelige orientering, vil gyroskopet holde treghetsformen. Friksjon i lufta vil etter tid bremse gyroskophjulet slik at momentet blir mindre, og akselen begynner å vingle for å prøve å opprettholde sin treghet. For at tregheten skal bevares må gyroskopet spinne med høy hastighet og massen må konsentreres mot kanten av rotoren.

Hvis vi implementerer denne type funksjonalitet inn i mikroelektroniske komponenter vil vi få en type lavkostprodukt som ofte blir kalt MEMS-gyroskop.

En tre-akset MEMS-gyro som er blir tatt hensyn til i dette prosjektet, benytter seg av akseenhetene rull(φ), stamp(θ) og kurs(ψ) som ble presentert i den matematiske delen for Eulervinklene. Ved rotering av denne MEMS-gyroen, finner man stillingen til gyroen med hensyn på tiden ved bruk av de matematiske grunnlagene for rotasjonsmatrisene i Eulervinklene. Igjen for denne oppgaven er det brukt beregnede måleverdier.



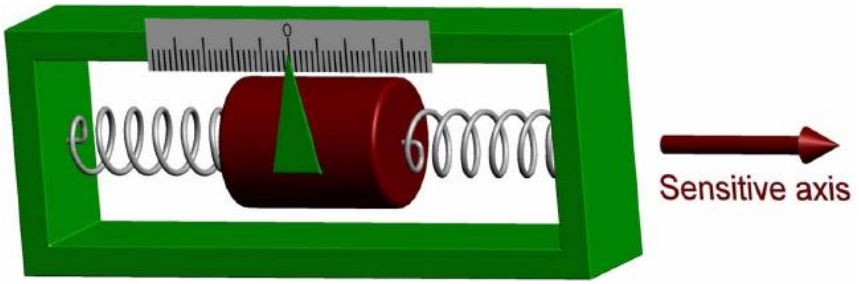
Figur 4.1.2: MEMS-gyroskop[34]

Sammen med et tre-dimensjonalt gyroskop benyttes også et MEMS-akselerometer av samme dimensjon. Dette er for å motta viktige verdier som gyroskopet ikke er i stand til å utgi, og gir nyttig tilleggsinformasjon for å beregne fartøyets bevegelse i fart.

4.1.2 Akselerometer

Et akselerometer er en innretning for måling av akselerasjon og krefter indusert av tyngdekraft[35]. Enkel- og fleraksemodeller er tilgjengelige for å detektere størrelsen og retningen på akselerasjonen som en vektorstørrelse. Akselerometre kan brukes for å måle helning, vibrasjon og støt.

For å illustrere funksjonaliteten til et akselerometer på en enkel måte, kan man feste en masse til en spiral. Hvis man utsetter denne for en horisontal bevegelse (x-retning) kan utslaget til denne massen leses av langs en målingsenhet.



Figur 4.1.3: Enkel fremstilling av et akselerometer og den funksjonalitet[36]

Sensorene brukes i økende grad i moderne elektronikk, mye innen for lavkost teknologi. Et akselerometer av denne typen kan produseres i singel-, dobbel- og trippel-akse design[37].

Figur 4.1.4: MEMS-akselerometer[38]

Kombinert med en MEMS-gyro utfyller disse to enhetene seg godt. Deres funksjonaliteter gjør at det er mulig å måle legemets fysiske parametere på en effektiv måte.

Ved å benytte et akselerometer med tre akser kan man måle den spesifikke kraften $\underline{f}^b$, og i tillegg måles også gravitasjonen med tanke på Einsteins prinsipp av ekvivalens.

4.2 Treghetsnavigasjon for kulejord

[39]Treghetsnavigeringen i denne oppgaven er gitt med en tre-akset plattform med roterende jord, hvor corioliskraften ikke er tatt hensyn til. Kulejord betyr at man tar hensyn til jordrotasjonen, samtidig som flyet beveger seg i banen. I denne delen er formen for det fysiske- og mekaniserte systemet med på å beskrive det tre-dimensjonale plattformen til TNS. I det mekaniserte systemet beskrives navigasjonslikningene med parametere fra det fysiske systemet.

4.2.1 Simuleringsmodell for det fysiske systemet

Banegeneratormodellen beskriver den sanne deterministiske systemmodellen for det fysiske systemet, hvor en *strek under* (_) viser til beregning av numerisk sann verdi. Fra generatoren er posisjon, hastighet og orientering til flyet beregnet med tanke på tiden. Tilstandsvektoren $\underline{x}$ for systemet inneholder dermed parameterne $\underline{p}_b^t$, $\underline{v}_b^t$ og Eulervinklene $\underline{\theta}$ som beskriver kursen til flyet. Tilstandsvektoren er gitt ved følgende for:

$$\underline{x} = \begin{bmatrix} \underline{p}_b^t \\ \underline{v}_b^t \\ \underline{\theta} \end{bmatrix} \tag{4.1}$$

Hvor initialiseringen er gitt ved tiden t_0:

$$\underline{p}_0^i, \underline{v}_b^i, \underline{\theta}(t_0) \tag{4.2}$$

Beregningene for spesifikk kraft og vinkelakslerasjon er gitt i koordinatsystemet til {b} relativt i ramme {i} og er også kjente fra banegeneratoren. Disse parameterne beskriver det sanne pådraget som systemet får inn, gitt med hensyn på tiden. Pådragsvektoren angis som en vektor av $\underline{u}(t)$ og er gitt på følgende form:

$$\underline{u}(t) = \begin{bmatrix} \underline{f}^b \\ \underline{\omega}_b^{ib} \end{bmatrix} \tag{4.3}$$

Hvor spesifikk kraft er beregnet på grunnlag av likningen (3.40), og vinkelhastigheten ved likning (3.33), samt på teorien i avsnitt 3.1.3.

4.2.2 Navigasjonslikninger (NAV)

Den mekaniserte systemdelen, også kalt navigasjon, beskriver TNS som en lineær deterministisk filtermodell. Navigasjonslikningene beregnes med bakgrunn på de fysiske verdiene. Ved å innføre integrasjonsalgoritmer beskrives de estimerte tilstandsverdiene for posisjon, hastighet og orientering ved {b} til {i} ved gitte likninger:

$$\dot{\underline{p}}_b^i = \underline{v}_b^i \tag{4.4}$$

$$\dot{\underline{v}}_b^i = \underline{f}^i - \underline{g}^i \tag{4.5}$$

$$\dot{\underline{\theta}} = D(\underline{\theta})\omega_b^t \tag{4.6}$$

Hvor $D(\underline{\theta})$ er definert av det kinematiske problem for 3-2-1 Eulervinkler og beskrevet i avsnitt 2.1.4.

Informasjonen navigasjonslikningene gir ut, ønskes å sammenliknes med de sanne verdiene fra det fysiske systemet. Med dette er det interessant å se om navigasjonslikningene kan brukes til å navigere flyet i riktig forhold til banen. Siden det ikke er implementert noe form for prosesstøy (w) eller målestøy (v) er dette sannsynlig hvis parameterne tilsies å være like. For best mulig sammenlikning må tilstandene transformeres fra {i} over til {t} systemet gitt fra {b}. Vinkelhastigheten $\underline{\omega}_b^t$, som også returneres fra banegeneratoren, blir beregnet på nytt ved bruka av $\underline{\omega}_b^{ib}$ og brukes for å angi orienteringen til navigasjonslikningene.

$$\underline{\omega}_b^t = R_b^t \underline{\omega}_b^{ib} - R_b^t R_i^b \underline{\omega}_e^i \tag{4.7}$$

Hastigheten gjøres om til formen:

$$\dot{\underline{v}}_b^t = R_b^t \underline{f}^b - \underline{g}^t - S\left(\underline{\omega}_e^i\right)\left\{S\left(\underline{\omega}_e^i\right)R_e^i\left[\underline{p}_t^e + R_t^e \underline{p}_b^t\right] + 2R_e^i R_t^e \underline{v}_b^t\right\} \tag{4.8}$$

Hvor posisjon finnes ved hjelp av integrasjonsalgoritmen og gir:

$$\dot{\underline{p}}_b^t = \underline{v}_b^t \tag{4.9}$$

Navigasjonslikningene for TNS systemet kan nå settes sammen på differensialformen, og ved de utledede likningene (4.7), (4.8) og (4.9) vil dette gi:

$$\underline{\dot{p}}_b^t = \underline{v}_b^t \tag{4.10}$$

$$\underline{\dot{v}}_b^t = R_b^t \underline{\tilde{f}}^b - \underline{g}^t - S\left(\underline{\omega}_e^i\right)\left\{S\left(\underline{\omega}_e^i\right)R_e^i\left[\underline{p}_t^e + R_t^e \underline{p}_b^t\right] + 2R_e^i R_t^e \underline{v}_b^t\right\} \tag{4.11}$$

$$\underline{\dot{\theta}} = D(\underline{\theta})\tilde{\omega}_b^t \tag{4.12}$$

Ved tiden t_0 vil tilstandene for navigasjonslikningene bli definert som

$$\underline{p}_0^t, \underline{v}_b^t, \underline{\theta}(t_0) \tag{4.13}$$

Pådraget som kjøres inn i TNS kommer opprinnelig fra IMU sensorene, som innledningsvis består av et akselerometer og en gyro. Fra disse enhetene er det vanlig å trekke ut kreftene og vinkelhastighetene som blir målt, men i dette tilfellet benyttes kun beregnede verdier. Den sanne spesifikke kraften $\underline{f}^b$ og vinkelhastigheten ω_b^t settes dermed som pådrag inn på systemet, hvor en *bølge* notasjon $(\sim)$ brukes til å henvise til "målte" verdier, men som likevel er beregnede. Dette gjør at forviklinger mellom verdier som brukes til et fysisk system og navigasjonslikningene kan unngås. Pådraget som genereres er gitt ved

$$\underline{u}(t) = \begin{bmatrix} \underline{\tilde{f}}^b \\ \underline{\tilde{\omega}}_b^t \end{bmatrix} \tag{4.14}$$

Siden det ikke er tatt med noe form for støy fra akselerometeret og gyroen, påvirkes ikke dette de sanne verdiene fra det fysiske systemet. Prosesstøy er også utlukket.

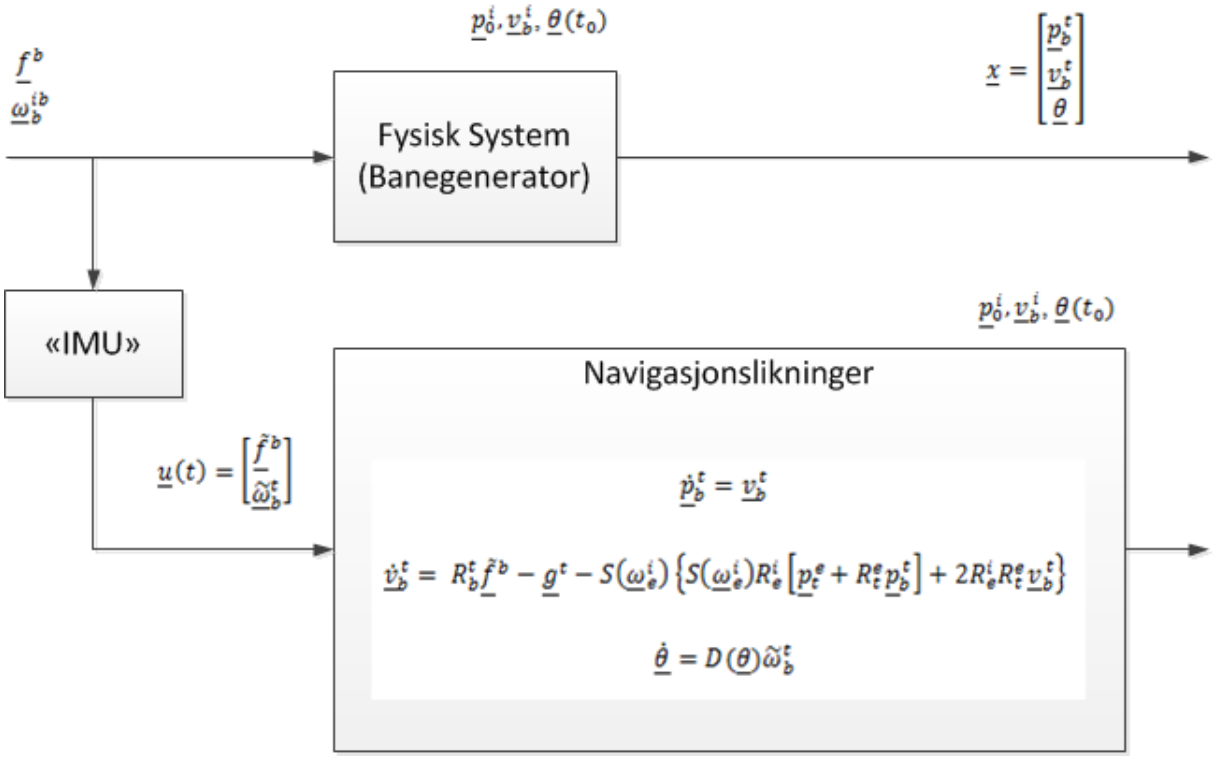
Det navigasjonssystemet kan nå settes på standardform og viser sammenhengen mellom tilstands- og pådragsegenskapene. Standardformen gir:

$$\underline{\dot{\tilde{x}}} = f\left(\underline{x}, \underline{u}\right) \tag{4.15}$$

$$\underline{\tilde{z}}_k = h * \left(\underline{x}_k\right) \tag{4.16}$$

4.2.3 Blokkskjema for TNS'et

Den matematiske modellen for navigasjonslikningene og det fysiske systemet i TNS'et kan oppsummeres med et blokkskjema. Dette blokkskjemaet illustrerer hvordan TNS fungerer for det som er blitt utført og gir en bedre fremstilling av hvilke verdier og tilstander som blir beregnet i riktig rekkefølge. I tillegg vises hva som blir sendt inn som pådrag og hva som sendes ut videre i systemet.



Figur 4.2.1: Oppsummering av TNS i et blokkskjema

4.2.4 Numerisk integrasjon for navigasjonslikningene

I navigasjonslikningene for TNS benyttes to typer metoder for integrasjon. Først Euler's metode, etterfulgt av Heun's metode[40]. Heun's metode blir ofte kalt for Runge-Kuttas metode og er av 2.orden. Begge metodene omgjør derivasjonsleddene til tilstandene og pådraget til form av (k+1) og integrasjonssekvensene blir enklere og prediktere ved addisjonsregning. Metodene brukes også fordi endringen til tiden ikke forandres konstant, slik at Δt vil variere ettersom hvor flyet befinner seg.

Euler's metode baserer seg på nåværende punkt x_k til neste predikterte punkt x_{k+1}, punktet evalueres og adderes med det tidligere punktet x_k. Videre tar man gjennomsnittet av

det nåværende punktet og det predikterte ved å innføre et multiplikasjonsledd Δt for å så bruke denne til å bevege seg til neste punkt. Likningen for Euler's metode er gitt ved likning:

$$\underline{x}'_{k+1} = \underline{x}_k + h * f(\underline{x}_k, \underline{u}_k) \tag{4.17}$$

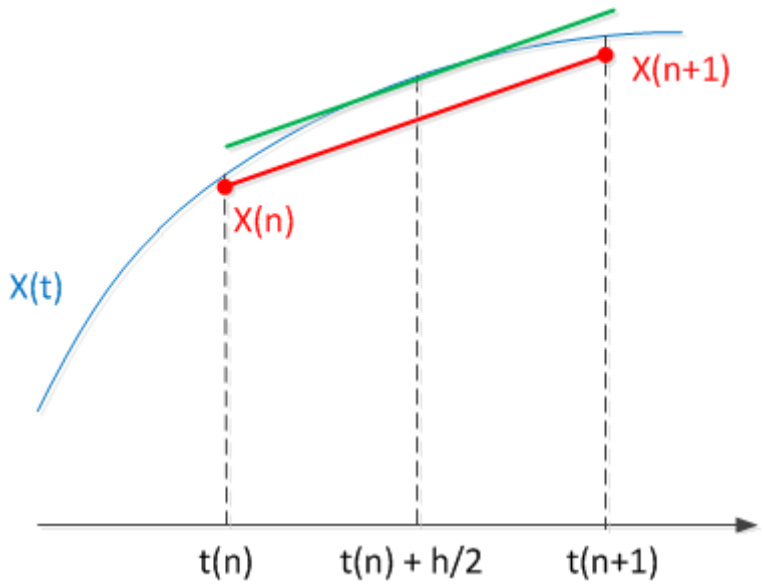
Hvor

$$h = \Delta t \tag{4.18}$$

Heun's metode beregner også nåværende og predikterte punktverdier, men i tillegg brukes de estimerte verdiene fra Euler's og legges til nåværende punkt av x_k. Gjennomsnittsberegningen for denne metoden gjør at oppløsningen mellom sekvensene blir mer nøyaktig, og beregnes ved å multiplisere $(\Delta t/2)$. Dette foretar dobbelt så mange funksjonsbetraktninger som Euler, men minking av feil etterfulgt av proporsjonalitet med 2.orden, derfor er ikke dette et dårlig vurderingsvalg av metode. Likningen for Heun's metode er gitt ved:

$$\underline{x}_{k+1} = \underline{x}_k + \frac{h}{2}\left(f(x_k, \underline{u}_k) + f(\underline{x}'_{k+1}, \underline{u}_{k+1})\right) \tag{4.19}$$

Med disse metodene vil navigasjonslikningene for det mekaniserte systemet få god nøyaktighet.



Figur 4.2.2: Diskretisering ved Heun's metode basert på Euler's

5 Simuleringsresultater

Fremgangsmåten og beregninger som er gjort rede for, er testet opp og simulert i henhold til begrunnet teoretisk bakgrunn, og vil vise resultatene for hele systemet som er blitt implementert med bestemte algoritmer i MATLAB. De ulike delene som fremstiller resultatene, viser grafiske figurer som beskriver tilstandene som mottas fra banegeneratoren og TNS. For å motta figurene, kjøres hovedfilen `Main.m` i MATLAB som kaller på funksjonen `Simuleringsresultater.m`.

En numerisk feil som kommer fra banegeneratoren er tatt hensyn til, og er med på å lage høye "*spikere*" i grafene. Dette kommer av feil estimering mellom avstanden til hvert samplingspunkt. Denne spesielle hendelsen inntreffer når flyet når sitt tangentpunkt i hver sving, og klarer ikke å beregne riktig i forhold til referansepunktet. Dette fører til at avstanden mellom samplingspunktene ikke blir like og nøyaktige, og er grunnen til at Euler's og Heun's metode er benyttet for å integrere opp navigasjonslikningene i TNS. Dette viser også til hvorfor definisjonen (4.18) brukes ved integrasjonsmetodene.

Grafene for de enkelte figurene er fremstilt med x- og y-akser, hvor hver akse har sin definerte beskrivelse. Akse y gjengir flyets tilstand i $rad, rad/s, rad/s^2$ eller ved koordinatform $meter$. Akse x beskriver flyet med hensyn på tiden oppgitt i $sekunder$, eller ved koordinatform $meter$. Dette gjelder for alle simuleringsresultatene gitt videre og kan sees ut fra figurene.

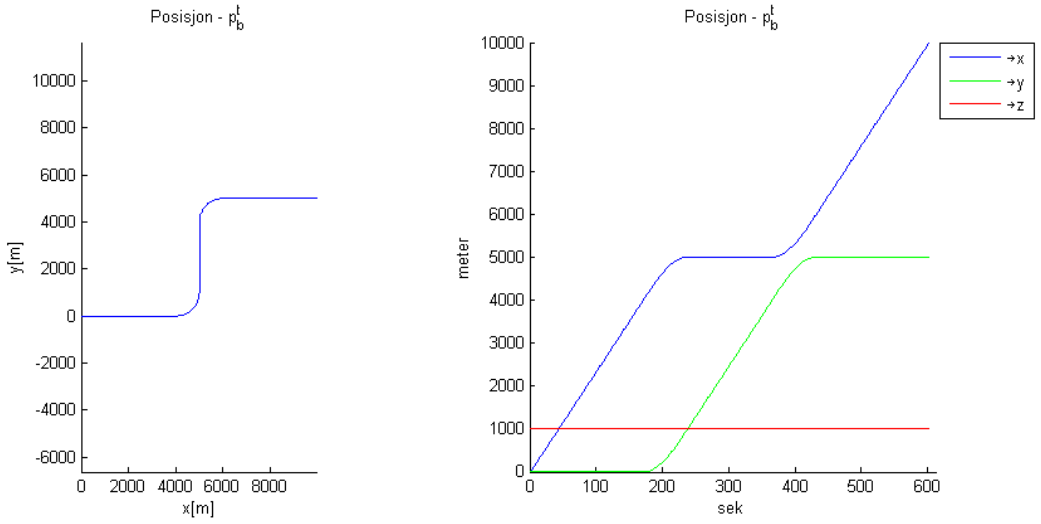
Hver figur vil bli beskrevet og diskutert i henhold til hva som er oppnådd, og enkelte vil bli begrunnet hvis eventuelle resultater ikke ble som ønsket.

5.1 Banegenerator

Banegeneratoren og den 2-dimensjonale kontrolleren, gir ut simuleringsresultater for flyets tilstandsverdier i {t}-rammen, samt informasjonen posisjon, hastighet og akselerasjon. Med disse resultatene, er det matematisk bevist at spesifikk kraft og vinkelhastighet kan beregnes ut fra generatoren. Kreftene og akselerasjonene som vil påvirke systemet i denne sammenhengen er vist i figur 5.1.6 og 5.1.7.

5.1.1 Posisjon $\left(\underline{p}_b^t\right)$

Simuleringsresultatet for flyets posisjon gjennom den definerte banen er:

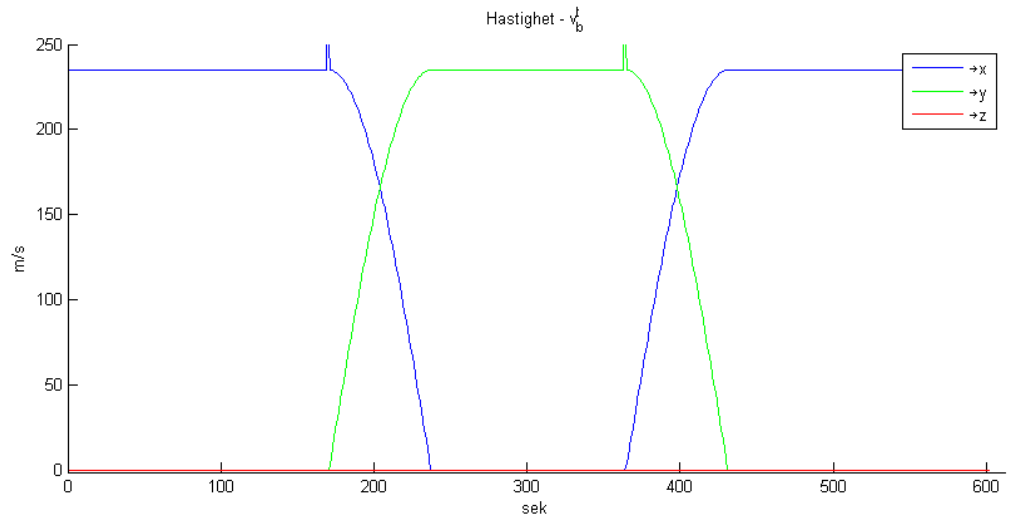


Figur 5.1.1: Simuleringsresultat for posisjon

Grafen til venstre i figur 5.1.1 fremstiller legemets posisjon og ved å sammenlikne denne grafen med figur 3.2.6, kan man se at det er store likheter mellom dem. Grafen er som ønsket. Grafen til høyre viser riktig verdi for x-, y- og z-retning sammenliknet med grafen til venstre. Langs en rett strekning øker x mens y holdes til konstant null, og motsatt når flyet skifter retning mot y-aksen. Høyden holdes konstant på 1000 meter over bakken.

5.1.2 Hastighet $(\underline{v}_b^t)$

Simuleringsresultatet for flyets konstante hastighet er:



Figur 5.1.2: Simuleringsresultat for hastighet

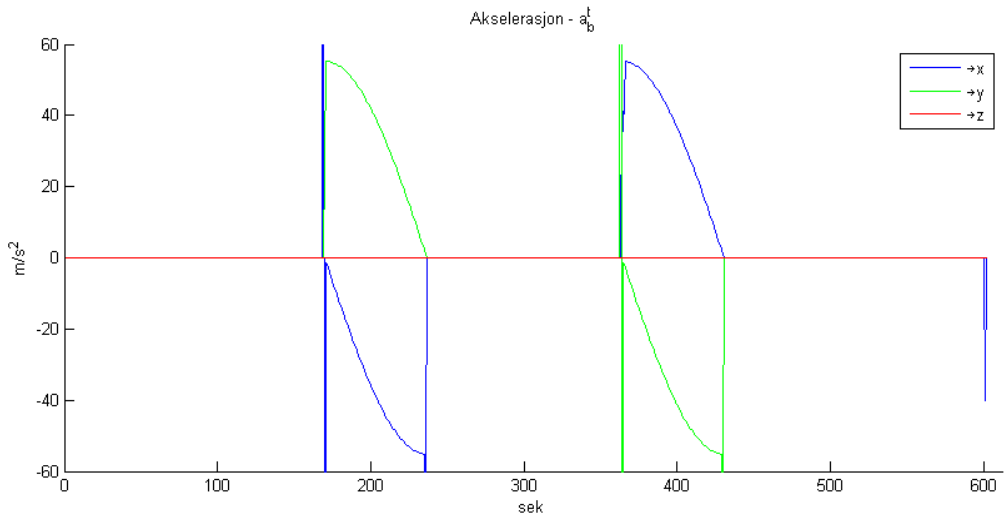
Figur 5.1.2 viser hvordan hastigheten forandrer seg i x- og y-retning referert til figur 5.1.1, og viser at flyet har en konstant hastighet på 235 m/s . Retning x mister pådraget sitt når en sving inntreffer og styres i y-retning. Hastigheten beveger seg ned til 0 m/s helt til flyet roteres og har en virkning tilbake. Følgende av dette gjør at hastigheten i y-retning øker etter første sving, og går fra 0 m/s til 235 m/s. Ved ny rotasjon reduseres pådraget tilbake til 0 m/s.

Avviket som ble referert til innledningsvis for resultatene, kan vises ved denne grafen. Når flyet når sitt første tangentpunkt, oppstår den numeriske feilen vist med en liten "*spiker*" i grafen. Den vil følgelig tas med videre i de andre resultatene, men feilen er såpass liten og vil ikke ha noen store påvirkninger for systemet ellers, dermed kan man se bort i fra dette.

Fra dette illustrerer figuren hastigheten som ønsket.

5.1.3 Akselerasjon $(\underline{a}_b^t)$

Simuleringsresultatet for flyets akselerasjon er:

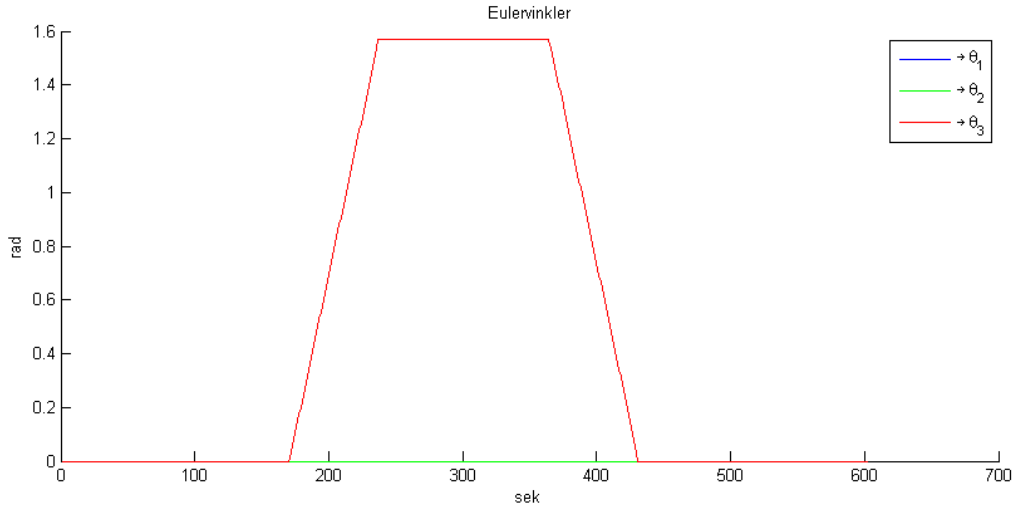


Figur 5.1.3: Simuleringsresultat for akselerasjon

Langs rette strekninger viser figuren 5.1.3 at det finnes ingen form akselerasjon i x-, y- og z-retning, men i svinger oppstår sentripetalakselerasjonen. I første sving får x en gradvis negativ påvirkning, mens y får en umiddelbar positiv påvirkning og reduseres gradvis ned. Det motsatte skjer i siste sving hvor flyet roterer tilbake til opprinnelig vinkelform og x får en umiddelbar positiv verdi, og reduseres igjen til enden av hver sving. Retning y får tilstanden til x ved første sving. Dette stemmer veldig godt overens med det faktiske systemet.

5.1.4 Orientering (Eulervinkler)

Simuleringsresultatet for orienteringen av systemet, beskrevet med Eulervinkler hvor kun θ_3 roterer, er vist ved:

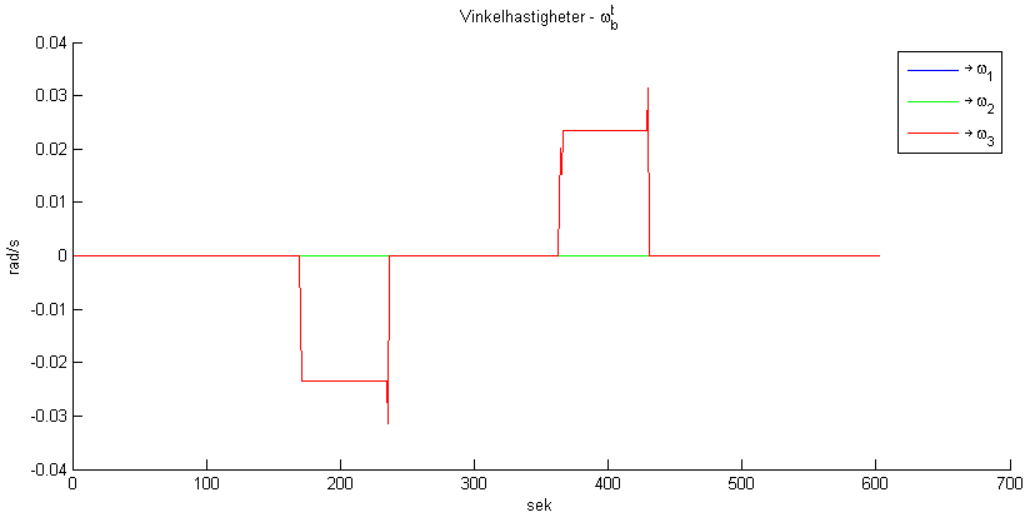


Figur 5.1.4: Simuleringsresultater for orientering ved bruk av Eulervinkler

Resultatet ved å bruke Eulervinkler i {t} rammen vises i figur 5.1.4 ved at flyet roterer positivt fra 0 til $\pi/2$ og negativt tilbake 0. Dette tilsvarer en rotasjon fra 0˚ til 90˚ ved første sving, og fra 90˚ til 0˚ etter andre sving. Komponentene x- og y-har ingen rotasjon. Sammenliknet med figure 5.1.1 stemmer rotasjonen med svingene i banen.

5.1.5 Vinkelhastighet i {t} rammen $(\underline{\omega}_b^t)$

Simuleringsresultatet for vinkelhastighetene i {t} rammen er gitt ved:

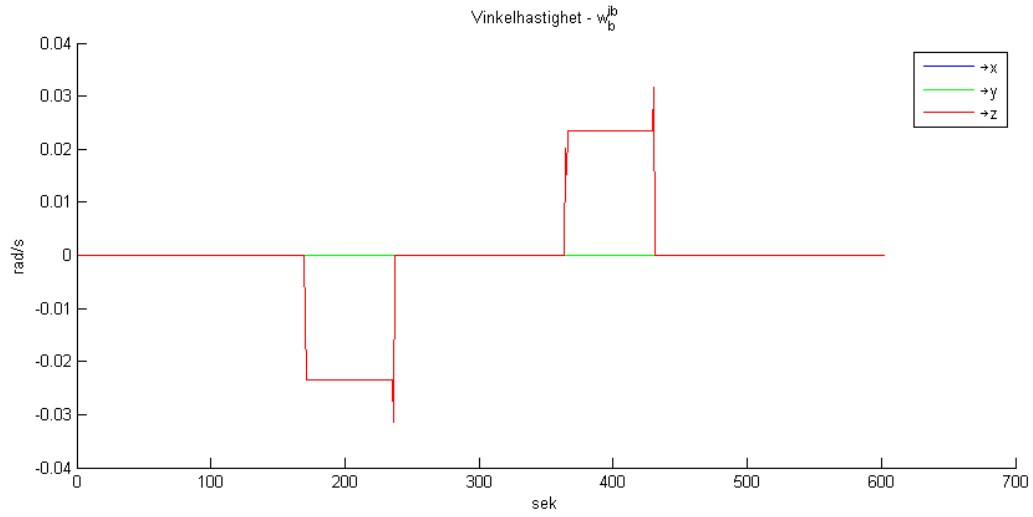


Figur 5.1.5: Simuleringsresultater for vinkelhastighet i {t} rammen

Vinkelhastigheten $\underline{\omega}_b^t$ er kun forskjellig fra null i z-retning i det flyet kjører gjennom hver sving. De to siste komponentene x og y, er alltid null ettersom tiden utvikler seg. Ved å derivere orienteringen av Eulervinklene, henvist i figur (x.x.x), vil vinkelhastighet være negativ og positiv gjennom hver sving.

5.1.6 Vinkelhastighet $\left(\underline{\omega}_b^{ib}\right)$

Simuleringsresultatet for vinkelhastighet $\underline{\omega}_b^{ib}$, som er beregnet med likning (3.33), er gitt ved

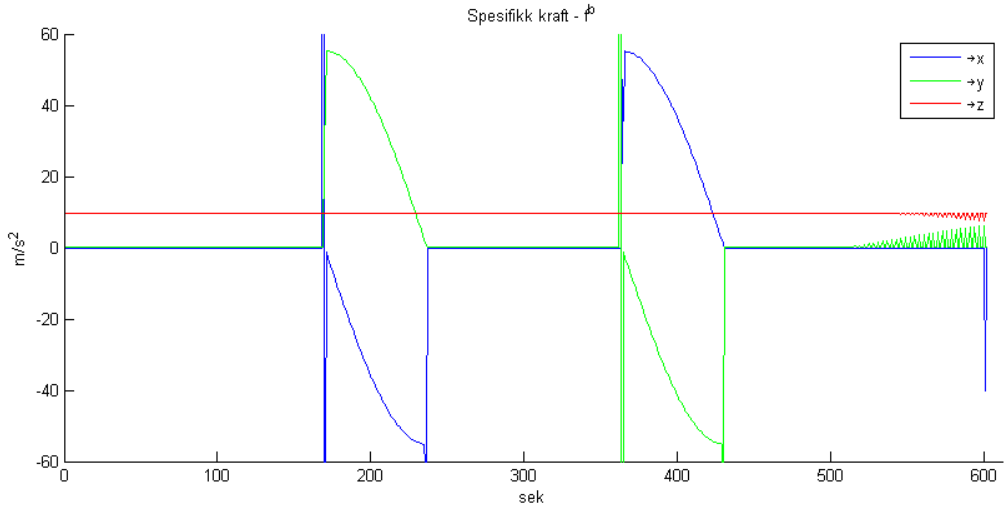


Figur 5.1.6: Simuleringsresultater for beregnet vinkelhastighet

Resultatet for vinkelhastigheten $\underline{\omega}_b^{ib}$ ønskes å ha tilnærmet like verdier som vinkelhastighet $\underline{\omega}_b^t$. Vinkelhastigheten i dette tilfellet har teoretisk ikke mange forskjellige påvirkningsfaktorer som endrer verdien fra {i} til {t}. Ved å sammenlikne figur 5.1.5 med figur 5.1.6 ser man at dette stemmer, og verdiene for denne beregningen er som ønsket. Komponenten z, er den eneste som endrer seg med tiden, mens x- og y-komponenten holdes til null.

5.1.7 Spesifikk kraft $\left(\underline{f}^{b}\right)$

Simuleringsresultatet for spesifikk kraft, som er beregnet med likning (3.40) er gitt ved:



Figur 5.1.7: Simuleringsresultater for beregnet spesifikk kraft

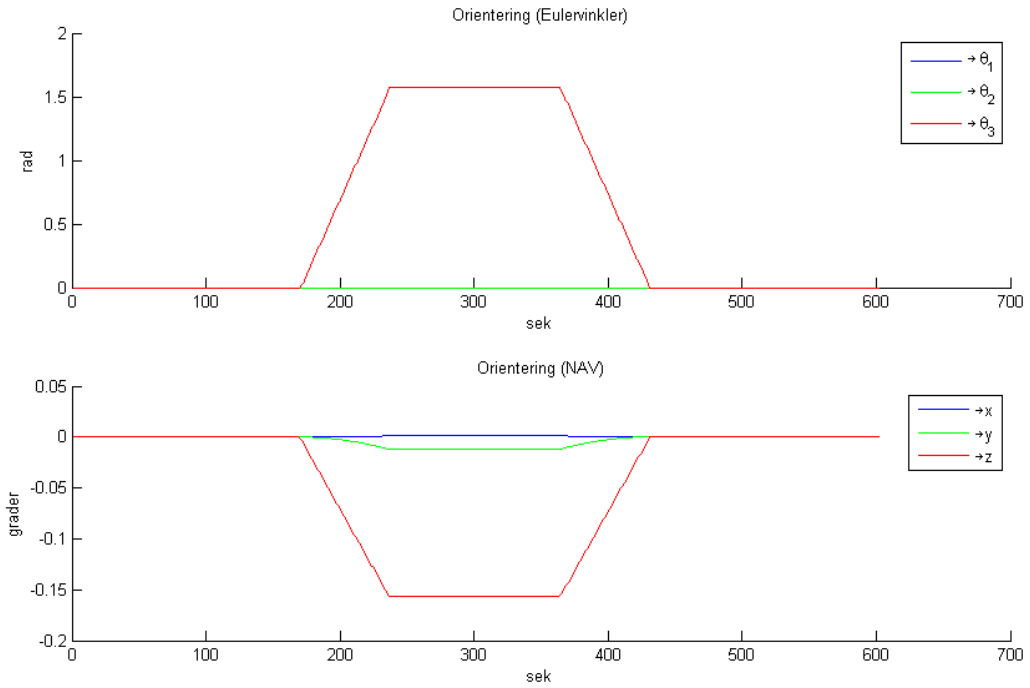
Ønsket spesifikk kraft som forventes av systemet og beregningen lagt til grunn for teorien, er samsvar med akselerasjonen som trekkes ut fra banegeneratoren. Ved å se på figur 5.1.7 og figur 5.1.3 kan det konkluderes med at disse er tilnærmet identiske og stemmer veldig godt overens med resultatet trukket ut fra flyets akselerasjon, med unntak fra de siste sekundene. Litt støy forplanter seg i verdiene og dette kan begrunnes med numeriske feil eller beregningsvariasjon med RKM og andre utregningsverdier som forplantes videre. I tillegg er gravitasjonskraften, som påvirker flyets tilstand i z-retning, med på å beskrive den spesifikke kraften. Dette kan også sees fra denne figuren sammenliknet med figuren for akselerasjon.

5.2 NAV

Simuleringsresultatene for navigasjon ønskes å være brukbare slik at de kan konkluderes på en logisk måte. Dette vil for eksempel være likhetstrekk som kan sammenliknes med de sanne tilstandsverdiene fra det fysiske systemet. For å kunne sammenlikne figurene best mulig, er figurene plassert med tilstandene fra generatoren og resultatet fra navigasjon i hvert tilfelle. Beskrivelse og kommentarer til figurene er gitt under hver figur og vil gi en forståelse på hvordan resultatene har oppstått.

5.2.1 Orientering (Eulervinkler) vs. Orientering (NAV)

Ved bruk av Heun's metode for navigasjonslikningene, blir simuleringsresultatet for orientering slik:



Figur 5.2.1: TNS - Simuleringsresultater for orientering (NAV)

Fra figur 5.2.1, sammenliknet med orienteringen fra Eulervinklene, har z-retningen ikke samme rotasjon i begge tilfeller. Navigasjonen gir ut en negativ orientering med mindre

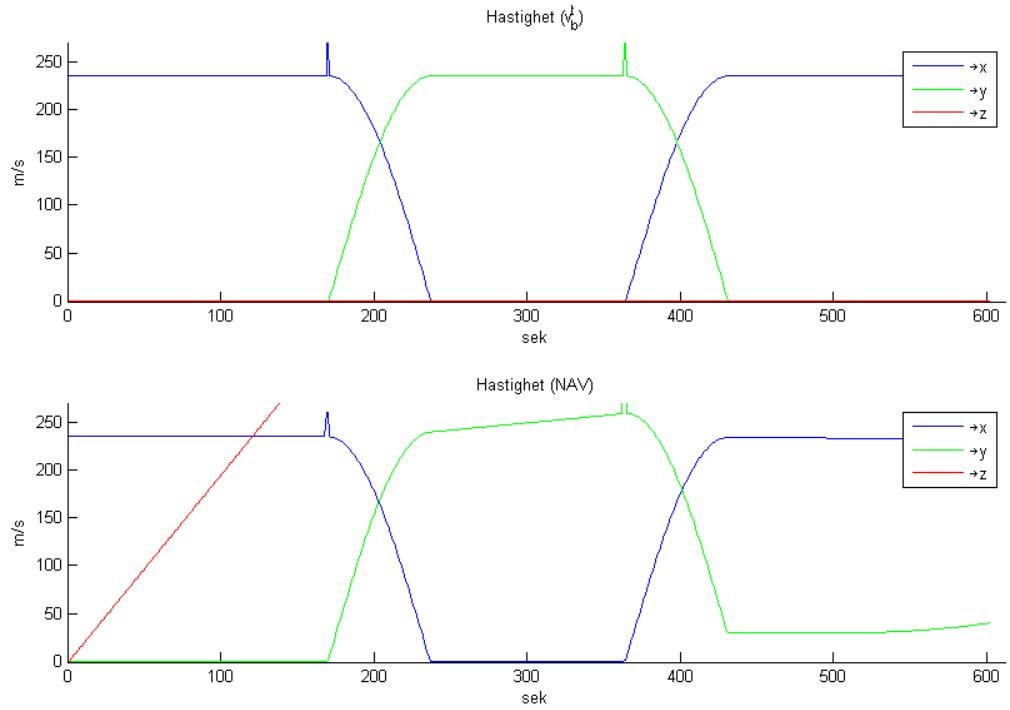
desimaler. Dette fører til at flyet ikke vil ha samme retning og plasserer flyet feil i henhold til det fysiske systemet.

Mulig feil som kan beskrive dette problemet er at integrasjonsalgoritmene legger ikke til en konstant verdi (C) som en matematisk regel for integrasjon. Tilføres en negativ konstant 10 i grafen, vil dette løfte grafen opp til positiv verdi og gir dermed riktig orientering oppgitt i rad/s. Problemet kan også ligge i programdelen, som tilsier at verdiene som brukes til å beregne orienteringen eller andre variabler, ikke blir hentet og brukt riktig. Et eksempel er å kalle på funksjoner som skal gi ut riktig verdi i et bestemt tilfelle. Denne følgefeilen vil dermed henge med videre og påvirke andre beregninger. Programmet som er laget for å estimere navigasjonslikningene er basert på teorien som er lagt til grunn for oppgaven, og ulike tester og feilsøkingsmetoder er brukt for å prøve å finne feil i programmet. Figurene viser dermed det beste resultatet som er oppnådd.

Siden denne feilen oppstår, kan det forventes at hastighet og posisjon også får en følgefeil.

5.2.2 Hastighet $(\underline{v}_b^t)$ vs. Hastighet (NAV)

Hastigheten for navigasjon, viser til simuleringsresultatet som følger:



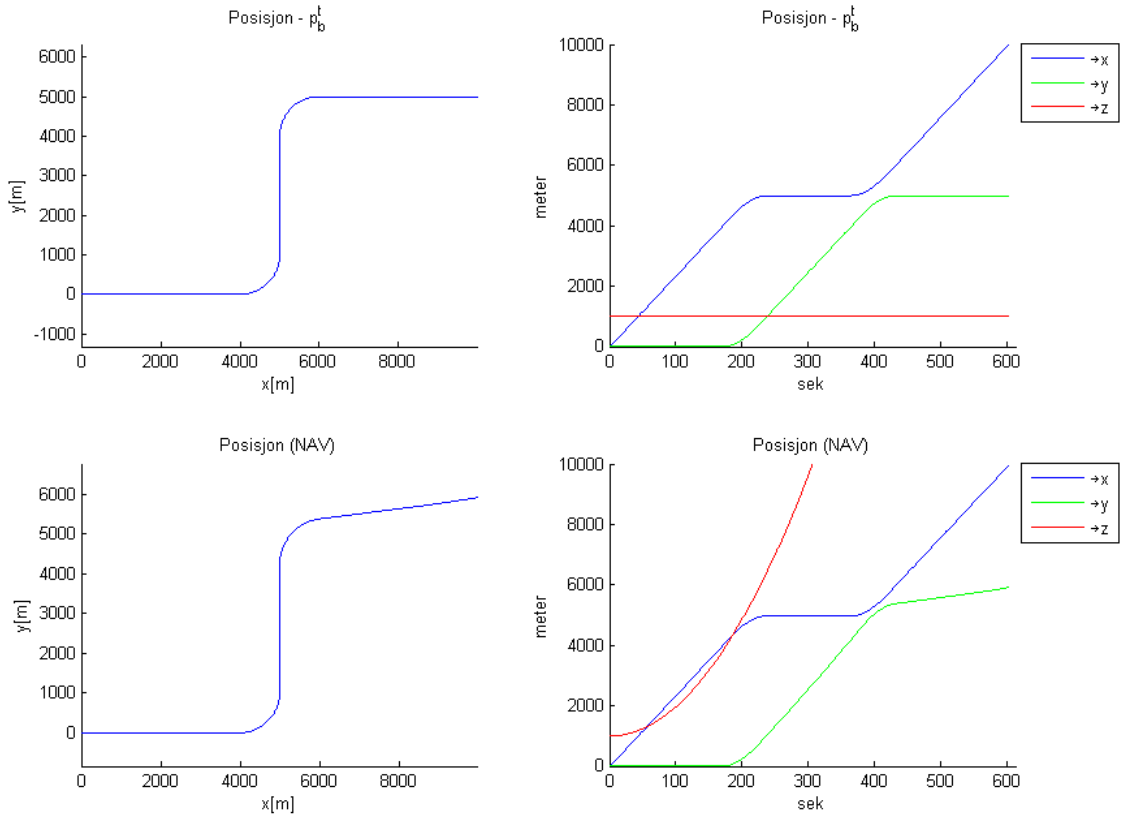
Figur 5.2.2: TNS - Simuleringsresultater for hastighet (NAV)

Figur 5.2.2 viser hastigheten i x- og y-retningen, sammenliknet med det fysiske systemet og her er det en fast korrelasjon mellom grafene. Siden orienteringen blir feil i forhold, er det tydelig at denne feilen detekteres videre i dette resultatet. Hastigheten langs y-retningen vil ikke stabiliseres tilbake til 0 etter den siste svingen er fullført, og vil føre til at flyet vil navigeres mer i y-retningen sammen med x-retningen. Hastigheten lang z-retningen stiger lineært og resulterer i at flyet vil øke i høyde ettersom tiden øker. Teoretisk anses dette som en type derivasjon, og kan sees ved å sammenlikne z-retningen i begge tilfeller.

For posisjon kan det forventes en eksponentiell stigning i z-retning og at hastigheten langs y-retningen vil påvirke hastigheten i x-retning lang den siste strekningen etter siste sving.

5.2.1 Posisjon ($\underline{p}_b^t$) vs. Posisjon (NAV)

Posisjonen for navigasjon, viser til simuleringsresultatet under:



Figur 5.2.3: TNS - Simuleringsresultater for posisjon (NAV)

Teoriantagelsene for posisjon, beskrevet i avsnitt 5.2.1, stemmer veldig godt med det som var forventet. Hastigheten i y-retningen etter siste sving forandrer posisjonen til flyet og plasserer det ut av kurs i forhold til det som er ønsket. Posisjonen i z-retning stiger eksponentielt og betyr at flyet forandrer høyde brått. Ønsket er at flyet skal ha en konstant horisontal høyde på 1000 meter, men ved gitt navigasjon vil flyet peker oppover og kjøre gjennom banen med en start høyde på 1000 meter og stige ettersom tiden utvikles. Dette er ikke særlig realistisk.

6 Konklusjon

Fremgangsmåte ved bruk av vektorer og sirkler for å realisere en banegenerator, har vist å gi et godt resultat. Banen blir av ønsket form etter hva brukeren av programmet setter som koordinatsposisjoner, og fremstiller en visuell figur som gir brukeren en oversikt over hvordan banen blir i forhold til de betingelsene som er bestemt. Hastighet og samplingstid kan også endres manuelt i programmet og kan bestemme tiden fra start til stopp.

Den 2-dimensjonale kontrolleren gir ut tilstandene for det fysiske systemet, og gir en avbildning av hvordan flyet kjører gjennom banen. Den numeriske feilen som oppstår kommer av unøyaktig sampling i forhold til tangentpunktene, og gjør at banen ikke blir særlig analytisk. Dette kan være en av grunnene til eventuelle følgefeil som blir tatt med i programmet og gjør figurene for simuleringsresultatene noe misvisende. Videre viser simuleringsresultatene at kontrolleren gir ut riktig informasjon om tilstandene og det fysiske systemet får ut en riktig tilstandsvektor til TNS.

Utledning av matematiske likninger og beregning av spesifikk kraft og vinkelhastighet, har resultert i forventede simuleringsresultater, og konkluderes med et akseptabelt pådrag inn på TNS.

Siden det ikke er laget et fullstendig TNS system for oppgaven, vil dette ha en forbindelse med at navigasjonen for flyet ikke blir som ønsket. For at hele TNS skal bli komplett, må det utledes bestemte feillikninger for systemet som beregner avviket mellom det fysiske systemet og navigasjonslikningene. Disse feillikningene beskrives som type sensorer som vurderer hver tilstand i begge tilfellene og regner ut forholdet mellom begge. Dette avviket brukes dermed videre til et lineært Kalmanfilter som filtrerer for predikterte feil og gir flyet mer stabilitet. Siden navigasjonslikningene ikke gir ut riktig verdier som ønsket, er dette en mulig løsning til å rette opp TNS. Kalmanfilteret vil forhåpentligvis gi riktig navigasjon i form av orientering, hastighet og posisjon som gjør at flyet kjører som ønsket. Dette kan være et eksempel på videre arbeid for oppgaven og er forklart nærmere i neste del.

Oppgaven kan til slutt konkluderes med at det over lang tid har blitt utviklet en deterministisk løsning for en banegenerator, som har gjort det mulig å realisere et fysisk system optimalt. Teorien som er studert, stemmer godt overens med utført arbeid og gjør det for eksempel mulig å beregne sanne verdier, i form av spesifikk kraft og vinkelhastighet inn

som et pådrag til et TNS. Siden oppgaven ikke har noe reelle måledata fra type MEMS gyro og akselerometer, er det greit å benytte samme pådrag inn til navigasjonslikningene. Oppgaven har vært veldig lærerik og arbeidet som er utført har gjort det mulig å simulere et TNS, bestående av et fysisk- og mekanisert system. Resultatene fra navigasjonslikningen gir mening etter utdyping i videre arbeid for oppgaven. Det er ikke bestemt at navigasjonen i TNS blir helt nøyaktig i samsvar med det fysiske systemet. Ved å filtrere bort feilen vil hele systemet fungere mer optimalt og resultatene vil bli bedre enn det som er oppnådd. Fra dette konkluderes det med akseptable resultater for utført arbeid.

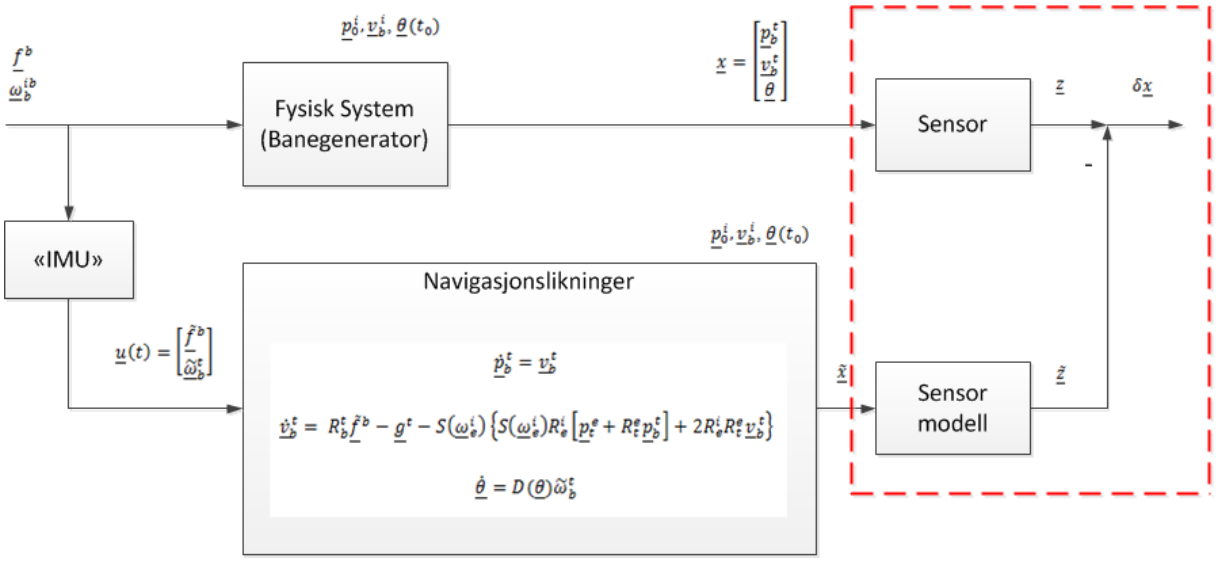
7 Videre arbeid

Videre arbeid som kan være aktuelt, er å finne en løsning på kontrolleren slik at den ikke begår noen form for numerisk feil, og gir ut nøyaktige verdier slik at banen blir mer analytisk.

Definering av feillikninger kan anses som type sensorer i TNS og gjør det helt komplett. Disse sensorene gir ut hver sin estimerte likning $\underline{z}_k$, beregner differansen mellom dem, og sendes videre til et kalmanfilter.

$$\delta \underline{z}_k = \underline{z}_k - \tilde{\underline{z}}_k \tag{7.1}$$

Tanken på hvordan TNS vil se ut etter dette arbeidet er vist i figuren under. Den ekstra delen som må legges til figur 4.2.1 er markert:



Figur 7.1.1: Utvidet TNS blokkskjema for videre arbeid

Siden feillikningen ikke har noe pådrag $\underline{u}$ vil denne bli satt til null.

Et lineært kalmanfilter kan bli satt opp og brukt som tilstandsestimator til det deterministiske systemet. Dette kan gjøres på grunnlag av differansefeil mellom posisjon-, hastighet- og orienteringsmålinger som mottas fra TNS, samt pådraget som sendes inn i systemet. Ved å anvende dette filteret vil en oppnå mer optimale estimater av navigasjonstilstandene og flyet vil fly mer presist, sammenliknet med det sanne fysiske systemet. Likningene for et tilbakekoblet filter vil ha formen:

$$\delta\dot{\underline{x}} = F(\tilde{\underline{x}}, t) \tag{7.2}$$

$$\delta\underline{z}_k = H(\tilde{\underline{x}}) \tag{7.3}$$

Hvis det lineære filteret ikke fungerer som ønsket, kan et utvidet kalmanfilter være av en bedre løsning.

Tilslutt kan man lage stokastiske feilmodeller for gyro og akselerometer som estimerer data med gitte parametere for flere nøyaktighetsklasser, og viser hvilke type MEMS sensorer som egner seg best for det bestemte systemet. Eventuelt en kovariansanalyse kan kjøres og testes opp mot de valgte nøyaktighetsklassene for å vurdere estimeringsfeil.

Referanser

[1] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s. 9

[2] Kenneth Gade, FFI – *Introduction to Inertial Navigation and Kalman filtering*, 09. Juni 2008. Figur s. 7

[3] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s. 13

[4] Beskrivelse av vektorlenge til et referanserom som kalles *norm* http://no.wikipedia.org/wiki/Norm_(matematikk) (18.05.2011)

[5] Beskrivelse av en geometrisk vektor som svarer til den *euklidiske* normen. http://no.wikipedia.org/wiki/Euklidsk_geometri (18.05.2011)

[6] Beskrivelse ved bruk av kryssprodukt http://nn.wikipedia.org/wiki/Kryssprodukt (18.05.2011)

[7] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s.14

[8] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s 15 – 16.

[9] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s. 17

[10] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer* Figur 2.1.2: Rotasjonssekvens 3-2-1 (Eulervinkler)

[11] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s. 18

[12] Dcmq – Own work. 2. februar 2010. http://en.wikipedia.org/wiki/File:Atan2_60.svg (18.05.2011)

[13] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s. 27

[14] Thor I. Fossen – *Navigasjonssystemer: matematisk modellering og estimering*. April 1998. s.27

[15] Figur 2.2.1: ECI-, ECEF-, og BODY-ramme. http://www.mathworks.com/help/toolbox/aeroblks/ecef6dof.gif (18.05.2011)

[16] Thor I. Fossen – *Navigasjonssystemer: matematisk modellering og estimering*.

April 1998. s.27 – 28.

[17] Thor I. Fossen – *Navigasjonssystemer: matematisk modellering og estimering*. April 1998. s.28

[18] Thor I. Fossen – *Navigasjonssystemer: matematisk modellering og estimering*. April 1998 s. 30

[19] Beskrivelse ved bruk av vektor beregning. http://no.wikipedia.org/wiki/Vektor_(matematikk) (18.05.2011)

[20] Johan Haugan – Formler og tabeller, 1. utgave, 9. opplag 2006. s. 60

[21] Beskrivelse av sentripetalakselerasjonsprinsippet og i hvilke tilfeller det gjelder. http://no.wikipedia.org/wiki/Sentripetalakselerasjon (18.05.2011)

[22] Johan Haugan – Formler og tabeller, 1. utgave, 9. opplag 2006. s. 61

[23] Beskrivelse av vinkelhastighet. http://en.wikipedia.org/wiki/Angular_velocity (19.05.2011)

[24] Beskrivelse av spesifikk kraft. http://en.wikipedia.org/wiki/Specific_force (18.05.2011)

[25] Oddvar Hallingstad – *Matematisk modellering av dynamiske systemer*, 17. april 2008. s. 30

[26] Beskrivelse av ekvivalensprinsippet. http://freak.no/forum/archive/index.php/t-179519.html (18.05.2011)

[27] Beskrivelse av CCW og CW http://en.wikipedia.org/wiki/Clockwise (18.05.2011)

[28] Beskrivelse av TNS http://translate.google.no/translate?hl=no&langpair=en|no&u=http://en.wikipedia.org/wiki/Inertial_navigation_system (18.05.2011)

[29] Bent-Andre Risnes – *Integrasjon av GPS og lavkost TNS for stillingsestimering*. 17.06.2002. s. 11

[30] Thor I. Fossen – *Navigasjonssystemer: matematisk modellering og estimering*. April 1998. s. 65

[31] Fordeler og ulemper med TNS utennom MEMS. http://www.caplex.no/Web/ArticleView.aspx?id=9338259 (18.05.2011)

[32] Beskrivelse på den klassiske varianetn til et gyroskop og deres funksjonalitet.

http://www.encyclopedia.com/topic/gyroscope.aspx (18.05.2011)

[33] Figur 4.1.1: Gyroskop
http://commons.wikimedia.org/wiki/File:3D_Gyroscope.png (18.05.2011)

[34] Figur 4.1.2: MEMS – gyroskop.
http://www.digi.no/844515/%ABgyromobiler%BB-kommer-for-fullt-i-2011
(18.05.2011)

[35] Beskrivelse av akselerometer.
http://no.wikipedia.org/wiki/Akselerometer (18.05.2011)

[36] Kenneth Gade, FFI – *Introduction to Inertial Navigation and Kalman filtering*, 09.
Juni 2008. Figur s. 13

[37] Beskrivelse av MEMS – akselerometer.
http://www.digoo.info/teknologi/2010/09/Hva-er-en-Akselerometer.html (18.05.2011)

[38] Figur 4.1.4: MEMS – akelerometer
http://www.eetasia.com/ART_8800547078_480500_NP_5415983b.HTM(25.05.2011)

[39] Oddvar Hallingstad – Eksempler på TNS modeller. 3. mars 2004

[40] Bruk av Euler's og Heun's metode.
http://en.wikipedia.org/wiki/Heun%27s_method (12.05.2011)

Vedlegg

A. Utledninger av likninger

A.1 Spesifikk kraft

Utledning av likninger for akselerasjon for ramme {b} sett fra ramme {i}, basert på likninger fra teorem A.19 fra [O. Hallingstad, 17.04.08, s.30]. $\underline{a}_b^i$ som brukes for å beregne $\underline{f}^b$:

$$\begin{aligned}\underline{p}_b^i &= \underline{p}_{ie}^i + \underline{p}_{et}^i + \underline{p}_{tb}^i \\ &= \underline{p}_e^i + R_e^i \underline{p}_t^e + R_t^i \underline{p}_b^t \\ &= R_e^i p_t^e + R_e^i R_t^e \underline{p}_b^t\end{aligned}$$

$$\begin{aligned}\underline{v}_b^i &= \dot{R}_e^i \underline{p}_t^e + R_e^i \dot{\underline{p}}_t^e + \dot{R}_e^i R_t^e \underline{p}_b^t + R_e^i \dot{R}_t^e \underline{p}_b^t + R_e^i R_t^e \dot{\underline{p}}_b^t \\ &= S\left(\underline{\omega}_e^i\right) R_e^i \underline{p}_t^e + \underline{0} + S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{p}_b^t + \underline{0} + R_e^i R_t^e \dot{\underline{p}}_b^t \\ &= S\left(\underline{\omega}_e^i\right) R_e^i \underline{p}_t^e + S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{p}_b^t + R_e^i R_t^e \underline{v}_b^t\end{aligned}$$

$$\begin{aligned}\underline{a}_b^i &= \dot{S}\left(\underline{\omega}_e^i\right) R_e^i \underline{p}_t^e + S\left(\underline{\omega}_e^i\right) \dot{R}_e^i \underline{p}_t^e + S\left(\underline{\omega}_e^i\right) R_e^i \dot{\underline{p}}_t^e + \dot{S}\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{p}_b^t + S\left(\underline{\omega}_e^i\right) \dot{R}_e^i R_t^e \underline{p}_b^t + \\ &\quad S\left(\underline{\omega}_e^i\right) R_e^i \dot{R}_t^e \underline{p}_b^t + S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \dot{\underline{p}}_b^t + \dot{R}_e^i R_t^e \underline{v}_b^t + R_e^i \dot{R}_t^e \underline{v}_b^t + R_e^i R_t^e \dot{\underline{v}}_b^t \\ &= \underline{0} + S\left(\underline{\omega}_e^i\right) S\left(\underline{\omega}_e^i\right) R_e^i \underline{p}_t^e + \underline{0} + \underline{0} + S\left(\underline{\omega}_e^i\right) S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{p}_b^t + \underline{0} + S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \dot{\underline{p}}_b^t + \\ &\quad S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{v}_b^t + \underline{0} + R_e^i R_t^e \dot{\underline{v}}_b^t \\ &= S\left(\underline{\omega}_e^i\right) S\left(\underline{\omega}_e^i\right) R_e^i \underline{p}_t^e + S\left(\underline{\omega}_e^i\right) S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{p}_b^t + S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{v}_b^t + S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{v}_b^t + \\ &\quad R_e^i R_t^e \underline{a}_b^t \\ &= S\left(\underline{\omega}_e^i\right) S\left(\underline{\omega}_e^i\right) R_e^i \underline{p}_t^e + S\left(\underline{\omega}_e^i\right) S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{p}_b^t + 2S\left(\underline{\omega}_e^i\right) R_e^i R_t^e \underline{v}_b^t + R_e^i R_t^e \underline{a}_b^t \\ &= S\left(\underline{\omega}_e^i\right) \left\{ S\left(\underline{\omega}_e^i\right) R_e^i \left[\underline{p}_t^e + R_t^e \underline{p}_b^t \right] + 2 R_e^i R_t^e \underline{v}_b^t \right\} + R_e^i R_t^e \underline{a}_b^t\end{aligned}$$

B. Pseudo kode

B.1 Simulering

SIMULERING

Function [$\underline{f}^b, \underline{\omega}_b^{ib}$] = Simulering ($Input\ variabler$)

% Initialisering av variabler

$N = Tid/\Delta t$

$g = 9{,}81$

$$\underline{g}^t = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$$\underline{g}^b = R_t^b \underline{g}^t = \begin{bmatrix} 0 \\ 0 \\ -9{,}81 \end{bmatrix}$$

$r_e = 6{,}370 * 10^6$

$$L = \frac{\pi}{3}$$

$$\underline{p}_t^e = [r_e \cos L\,; 0;\ r_e \sin L]$$

$$R_t^e = R(\theta_3, z)R(\theta_3, x - L) = \begin{bmatrix} 0 & -s(L) & c(L) \\ 1 & 0 & 0 \\ 0 & c(L) & s(L) \end{bmatrix}$$

$\omega_e = 7{,}292115e^{-5}$

$$\underline{\omega}_e^i = \begin{bmatrix} \underline{\omega}_{e,X}^i \\ \underline{\omega}_{e,Y}^i \\ \underline{\omega}_{e,Z}^i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 7{,}292115e^{-5} \end{bmatrix}$$

$$R_e^i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\omega_e * t_0) & -s(\omega_e * t_0) \\ 0 & s(\omega_e * t_0) & c(\omega_e * t_0) \end{bmatrix}$$

$$\underline{\omega}_t^e = \begin{bmatrix} \underline{\omega}_{t,X}^e \\ \underline{\omega}_{t,Y}^e \\ \underline{\omega}_{t,Z}^e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

For k = 1:N

% funksjon RKMRbt

Function R_b^t

$$R_b^t = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\varphi - s\psi c\varphi & c\psi s\theta c\varphi + s\psi s\varphi \\ s\psi c\theta & s\psi s\theta s\varphi + c\psi c\varphi & s\psi s\theta c\varphi - c\psi s\varphi \\ -s\theta & c\theta s\varphi & c\theta c\varphi \end{bmatrix}$$

end

$R_t^b = (R_b^t)^T$

$R_b^i = R_e^i R_t^e R_b^t$

$R_i^b = (R_b^i)^T$

$$R_e^i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\omega_e * t) & -s(\omega_e * t) \\ 0 & s(\omega_e * t) & c(\omega_e * t) \end{bmatrix}$$

$R_b^e = R_t^e R_b^t$

% funksjon Skjev

Function $S(\underline{\omega}_e^i)$ = Skjev($\underline{\omega}_e^i$)

$$S(\underline{\omega}_e^i) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

end

$\underline{p}_b^i = R_e^i p_t^e + R_e^i R_t^e \underline{p}_b^t$

$\underline{v}_b^i = S(\underline{\omega}_e^i) R_e^i \underline{p}_t^e + S(\underline{\omega}_e^i) R_e^i R_t^e \underline{p}_b^t + R_e^i R_t^e \underline{v}_b^t$

$\underline{a}_b^i = S(\underline{\omega}_e^i) \left\{ S(\underline{\omega}_e^i) R_e^i \left[\underline{p}_t^e + R_t^e \underline{p}_b^t \right] + 2 R_e^i R_t^e \underline{v}_b^t \right\} + R_e^i R_t^e \underline{a}_b^t$

$\underline{\omega}_b^{ib} = R_t^b \underline{\omega}_b^t + R_e^b \underline{\omega}_t^e + R_i^b \omega_e^i$

$\underline{f}^b = R_i^b \underline{a}_b^i + R_t^b \underline{g}^b$

end

$$\underline{u}(t) = \begin{bmatrix} \underline{f}^b \\ \underline{\omega}_b^{ib} \end{bmatrix}$$

end

B.2 Funksjon

FUNKSJON

Function f = funksjon($\underline{x}, \underline{u}, t$)

% Initialisering av variabler

$$\underline{p}_b^t = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\underline{v}_b^t = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix}$$

$$\underline{\theta} = \begin{bmatrix} x_7 \\ x_8 \\ x_9 \end{bmatrix}$$

$$\underline{f}^b = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\underline{\omega}_b^{ib} = \begin{bmatrix} u_4 \\ u_5 \\ u_6 \end{bmatrix}$$

$$r_e = 6{,}370 * 10^6$$

$$\underline{p}_t^e = [r_e \cos L\,; 0;\ r_e \sin L]$$

$$\omega_e = 7{,}292115e^{-5}$$

$$\underline{\omega}_e^i = \begin{bmatrix} \underline{\omega}_{e,X}^i \\ \underline{\omega}_{e,Y}^i \\ \underline{\omega}_{e,Z}^i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 7{,}292115e^{-5} \end{bmatrix}$$

$$g = 9{,}81$$

$$\underline{g}^t = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

$$R_t^e = R(\theta_3, z)R(\theta_3, x - L) = \begin{bmatrix} 0 & -s(L) & c(L) \\ 1 & 0 & 0 \\ 0 & c(L) & s(L) \end{bmatrix}$$

$$R_e^i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\omega_e * t) & -s(\omega_e * t) \\ 0 & s(\omega_e * t) & c(\omega_e * t) \end{bmatrix}$$

% funksjon RKMRbt

Function R_b^t

$$R_b^t = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\varphi - s\psi c\varphi & c\psi s\theta c\varphi + s\psi s\varphi \\ s\psi c\theta & s\psi s\theta s\varphi + c\psi c\varphi & s\psi s\theta c\varphi - c\psi s\varphi \\ -s\theta & c\theta s\varphi & c\theta c\varphi \end{bmatrix}$$

end

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \underline{v}_b^t$$

$$\begin{bmatrix} f_4 \\ f_5 \\ f_6 \end{bmatrix} = R_b^t \underline{\tilde{f}}^b - \underline{g}^t - S\left(\underline{\omega}_e^i\right)\left\{S\left(\underline{\omega}_e^i\right) R_e^i \left[\underline{p}_t^e + R_t^e \underline{p}_b^t\right] + 2R_e^i R_t^e \underline{v}_b^t\right\}$$

$$\tilde{\omega}_b^t = R_b^t \underline{\omega}_b^{ib} - R_t^e R_e^i \underline{\omega}_e^i$$

Function $D(\underline{\theta})$

$$D = \begin{bmatrix} 1 & s\theta_1 t\theta_2 & c\theta_1 t\theta_2 \\ 0 & c\theta_1 & -s\theta_1 \\ 0 & \frac{s\theta_1}{c\theta_2} & \frac{c\theta_1}{c\theta_2} \end{bmatrix}$$

end

$$\begin{bmatrix} f_4 \\ f_5 \\ f_6 \end{bmatrix} = D(\underline{\theta})\tilde{\omega}_b^t$$

end

B.3 Navigasjon

NAV

Function $\tilde{\underline{x}}$ = funksjon($\underline{x}, \underline{u}, t$)

% Initialisering av variabler

$$\underline{x}_0 = \begin{bmatrix} \underline{p}_b^t(t_0) \\ \underline{v}_b^t(t_0) \\ \underline{\theta}(t_0) \end{bmatrix}$$

For k = 1:N-1

$$h = \Delta t$$

$$\underline{u}_k = \begin{bmatrix} \tilde{\underline{f}}^b(t_k) \\ \tilde{\underline{\omega}}_b^t(t_k) \end{bmatrix}$$

$$\underline{u}_{k+1} = \begin{bmatrix} \tilde{\underline{f}}^b(t_{k+1}) \\ \tilde{\underline{\omega}}_b^t(t_{k+1}) \end{bmatrix}$$

$$\underline{x}'_{k+1} = \underline{x}_k + h * \text{funksjon}(\underline{x}, \underline{u}, t)$$

$$\underline{x}_{k+1} = \underline{x}_k + \frac{h}{2}\Big((\text{funksjon}(\underline{x}, \underline{u}, t)) + \big(\text{funksjon}(\underline{x}'_{k+1}, \underline{u}_{k+1}, t_{k+1})\big)\Big)$$

end

$\tilde{\underline{x}} = \underline{x}$

end

C. Programmerings kode

C.1 Main.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               MASTEROPPGAVE                   %
%               UNIK Vår 2011                   %
%                                               %
%            HOVEDPROGRAM - MAIN                %
%                                               %
%       Kenneth Holterhuset Johansen            %
%                 ELD 5930                      %
%                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear all; clf; close all;
%% Initialisering

P1 = [0 0];                 % Posisjon 1
P2 = [5 0];                 % Posisjon 2
P3 = [5 5];                 % Posisjon 3
P4 = [10 5];                % Posisjon 4

VB = [P1; P2; P3; P4]*1000; % Veibane matrise

Radius = 1000;              % Sving radius til banen oppgitt i meter (m)
fart = 235;                 % Hastighet til flyet (m/s)
Ts = 0.1;                   % Samplings konstant
                            % Samplingslengden blir med gitt fart, lengde
                            % og samplingskonstant: 60 sekunder

%% BANEGENERATOR
[TP1,TP2,Origo,Bue]=Banegenerator(Radius,VB);

%% TRACKBANE
[p_b_t,Tid,v_b_t,a_b_t,Vinkel,w_b_t,Vinkelakselerasjon,x] =
TrackBane(VB,TP1,Origo,Bue,Radius,Ts,fart);

%% SIMULERING (f_b, w_b_ib)
[f_b,w_b_ib,u] = Simulering(p_b_t,Tid,v_b_t,a_b_t,Vinkel,w_b_t);

%% NAV
xBolge = NAV(x,u,Tid);

%% SIMULERINGSRESULTATER (PLOT/FIGURER)
Simuleringsresultater(p_b_t,v_b_t,a_b_t,f_b,w_b_ib,Vinkel,w_b_t,Vinkelaksel
erasjon,xBolge);
```

C.2 Banegenerator.m

```
function[TP1,TP2,Origo,Bue]=Banegenerator(Radius,VB)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               MASTEROPPGAVE                       %
%               UNIK Vår 2011                       %
%                                                   %
%         Funksjon - BANEGENERATOR                  %
%                                                   %
%       Kenneth Holterhuset Johansen                %
%                ELD 5930                           %
%                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %% ESTIMERING AV FLYBANE
    for i = 1:2
        %% Plot veibane punkter
        figure(1)
        plot(VB(:,1)',VB(:,2)','black--o','LineWidth',1);
        title('F L Y B A N E');
        axis equal
        xlabel('x[m]')
        ylabel('y[m]')

        %% Vektorer beregning til veibane
        A_ = VB(i,:)-VB(i+1,:);     % Vektor A
        B_ = VB(i+2,:)-VB(i+1,:);   % Vektor B

        % Sirkel variabler
        alpha = 0.5*acos(((A_)*(B_)')/(norm(A_)*norm(B_)));
        delta = tan((pi/2)-alpha)*Radius;

        % Enhetsvektor A
        EnhetsVekA = A_/norm(A_);    % A/||A||
        EnhetsVekB = B_/norm(B_);    % A/||A||

        %% Sirkler
        % Tar kryssproduktet til A&B vektor for definere sirkel på riktig
        % side av banen
        Kryssprod = A_(1)*B_(2)-A_(2)*B_(1);

        if Kryssprod<0
            % Ortogonalvektor: Definerer om sirkel er på riktig side av
              VB
            OrtoVek = [EnhetsVekA(2),-EnhetsVekA(1)];
        else
            OrtoVek = [-EnhetsVekA(2),EnhetsVekA(1)];
        end

        % Finne punktet til Origo
        Origo(i,:) = VB(i+1,:)+(EnhetsVekA*delta)+(OrtoVek*Radius);

        % Kaller på plotsirkel funksjonen som lager sirkel 1 & 2
        plotsirkel(Radius,Origo(i,:));

        %% Tangentpunkter
```

```
        % Tangentpunkt 1 i sirkel 1 & 2
        TP1(i,:) = VB(i+1,:)+(EnhetsVekA*delta);

        % Tangentpunkt 2 i sirkel 1 & 2
        TP2(i,:) = VB(i+1,:)+(EnhetsVekB*delta);

        % Plotter tangentpunktene i sirkel 1 & 2
        plot(TP1(i,1),TP1(i,2),'p');
        plot(TP2(i,1),TP2(i,2),'p');

        %% Sirkelbuer
        % Definerer Bue 1 & 2
        Bue(i,1) = atan2(TP1(i,2)-Origo(i,2),TP1(i,1)-Origo(i,1));
        Bue(i,2) = atan2(TP2(i,2)-Origo(i,2),TP2(i,1)-Origo(i,1));

        % Kaller på plotbue funksjon som plotter Bue 1 & 2
        plotbue(Radius, Origo(i,:), Bue(i,:))

        %% Vektor lengder
        % Lag vektor linjer ved å kalle på plotlengde funksjon
        plotlengde(i, VB, TP1, TP2)

    end
end
```

C.2.1 Plotsirkel.m

```
function plotsirkel(Radius,Origo)
%% Plotter sirkelbuer med radius og origo punkter

Ts = 0.001;                 % Sampling konstant for sirkler
t=0:Ts:2*pi;                % Sampling pr. punkt i en sirkel O=2*pi
x=Origo(1)+Radius*cos(t);   % x-koordinater for sirkelen
y=Origo(2)+Radius*sin(t);   % y-koordinater for sirkelen

hold on,                    % Hold sirkelform
% plot(x,y,'r');            % Plot x & y koordinatene, Farge: Rød

end                         % ##FUNKSJON SLUTT##
```

C.2.2 Plotbue.m

```
function plotbue(Radius, Origo, Bue)
%% Plotter sirkelbuer til veibanen med radius og origo punkt

    if Bue(1)<Bue(2)

        % Sirkelretning er CCW
        t=Bue(1):0.001:Bue(2);

    elseif Bue(1)>Bue(2)

        % Sirkelretning er CW
        t=Bue(1):-0.001:Bue(2);

    end

% Sirkel med [x,y]
x=Origo(1)+Radius*cos(t);
y=Origo(2)+Radius*sin(t);

hold on,                                % HOLD ON: ikke reset
plot(x,y,'m','LineWidth',2);            % Plotter sirkelbue

end
```

C.2.3 Plotlengde.m

```
function plotlengde(i, VB, TP1, TP2)
%% Lag vektor linjer

LV = [0 0; 0 0];            % Linje vektor
LV2 = [0 0; 0 0];           % Linje Vektor 2

    if i==1

        % Legger P1 og TP1(1) i LV matrise
        LV(:,:)=[VB(i,:);TP1(i,:)];
        hold on;

        % Plotter Linjevektor 1
        plot(LV(:,i),LV(:,i+1),'k','LineWidth',2)

    elseif i==2

        % Legger TP2(1) og TP1(2) i LV2 matrise
        LV2(:,:)=[TP2(i-1,:);TP1(i,:)];
        hold on;

        % Plotter Linjevektor 2
        plot(LV2(:,i-1),LV2(:,i),'k','LineWidth',2)

        % Legger TP2(2) og P4 i LV matrise
        LV(:,:)=[TP2(i,:);VB(i+2,:)];
        hold on;

        % Plotter siste linjevektor (spesial tilfelle)
        plot(LV(:,i-1),LV(:,i),'k','LineWidth',2)

    end

end
```

C.3 TrackBane.m

```
function [p_b_t,Tid,v_b_t,a_b_t,Vinkel,w_b_t,Vinkelakselerasjon,x] =
TrackBane(VB,TP1,Origo,Bue,Radius,Ts,fart)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               MASTEROPPGAVE                   %
%               UNIK Vår 2011                   %
%                                               %
%          Funksjon - TRACKBANE                 %
%                                               %
%       Kenneth Holterhuset Johansen            %
%                 ELD 5930                      %
%                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Variable initialisering
    i=1;                        % i = Matrise verdi i variabler
    j=2;                        % j = Sampel fremover i det fysiske systemet

    p_b_t(1,:) = VB(1,:);       % Start verdi på posisjon: P1
    Tid = 0;                    % Tid starter på 0
    TsLinje = Ts;               % Samplingkonstant for rett linje
    TsSirkel = Ts;              % Samplingkonstant for sirkel

    mode=1;                     % Mode: Linje
    run=1;                      % Kjør while 1 = run

while run
    %% Tracking langs en rett linje

    % mode==1 kjører kun når en rett linje inntreffer
    if mode==1;

        % Tid pr. sampling
        Tid(j)=Tid(j-1)+TsLinje;

        s(j) = fart*TsLinje;        % s = Strekning

        % Lengde på rett linje
        Temp=VB(i+1,:)-VB(i,:);

        % Beregning av vinkel retning
        vinkel = atan2(Temp(1,2),Temp(1,1));

        % Rotasjonsmatrise
        Rot = [cos(vinkel), -sin(vinkel);
               sin(vinkel), cos(vinkel)];

        % Posisjonsstilling pr. sampel lagres m/hensyn på endring i
          vinkel
        p_b_t(j,:)=p_b_t(j-1,:)+(Rot*[s(j);0])';

        % Plot Tracking langs rett linje
        hold on
%         plot(p_b_t(j,1),p_b_t(j,2),'ro');
```

```
        % Tegn samplene med engang
        drawnow

        if i<3

            % Når ønsket punkt er nådd, beregn ny retning
            if norm(TP1(i,:)-p_b_t(j,:))<(23.1)

                TsLinje2=(norm(TP1(i,:)-p_b_t(j-1,:)))/fart;

                s(j) = fart*TsLinje2;

                p_b_t(j,:)=p_b_t(j-1,:)+(Rot*[s(j);0])';

                % Skift til mode=2 'Bue'
                mode=2;

                % Initialiserer BuePos
                BuePos = 0;

                % Sampling for sirkel
                TsSirkel = Ts;
            end

        % Ellers slutt tracking når siste
        elseif norm(VB(4,:)-p_b_t(j,:))<(23.1)

            % veibane punkt er nådd
            run=0;

        end

        % Øk med et skritt fremover
        j=j+1;

    end % END if mode==1
    %% Tracking av sirkel

        % mode==2 kjører kun når en bue inntreffer
        if mode==2

        % Tid pr. sampel
        Tid(j) = Tid(j-1)+TsSirkel;

        s(j) = fart*TsSirkel;        % s = Strekning

           if Bue(i,1)<Bue(i,2)

                % Beregninger positiv vinkel til Bue
%                 BuePos = BuePos+BueVinkel;
           BuePos = BuePos+(s(j)/Radius);
                if ((Bue(i,1)+BuePos)>Bue(i,2))

                    BuePos = BuePos-(s(j)/Radius);

                    s(j)=fart*TsSirkel;
```

```
                    % Skift til mode 1 når Bue er feridg
                    mode=1;
                end

            elseif Bue(i,1)>Bue(i,2)

                % Beregninger negativ ivinkel til Bue
                BuePos = BuePos-(s(j)/Radius);

                if ((Bue(i,1)+BuePos)<Bue(i,2))

                    BuePos = BuePos+(s(j)/Radius);

                    s(j) = fart*TsSirkel;

                    % Skift til mode 1 når Bue er feridg
                    mode=1;
                end
            end

            if mode==1

            % Hvis mode=1 intreffer, øk i
                i=i+1;

            else

                % Plot tracking av Bue
                Tid(j)=Tid(j-1)+TsSirkel;
                x = Origo(i,1)+Radius*cos(Bue(i,1)+BuePos);
                y = Origo(i,2)+Radius*sin(Bue(i,1)+BuePos);
                mot=Origo(i,1)-x;
                hos=Origo(i,2)-y;
                posvinkel=atan2(mot,hos);
                posvinkel=atan2(hos,mot);

                p_b_t(j,:) = [x,y];
%                 plot(p_b_t(j-1,1),p_b_t(j-1,2),'ro');
                drawnow
                j=j+1;

            end % END if 'Bue' mode=1

        end % END if mode==2

    end  %END WHILE


%% Beregning av vinkel endring for flyets posisjon, hastighet og
akselerasjon

    % Vinkel endring for posisjon (f(x))
    for j=1:length(p_b_t(:,1))-1
         Est = p_b_t(j+1,:)-p_b_t(j,:);
         Vinkel(j) = (atan2(Est(2),Est(1)));
    end
```

```
    % Vinkel endring for hastighet (df/dx)
    for j=1:length(Vinkel)-1
         w_b_t(j) = Vinkel(j)-Vinkel(j+1);
    end

    % Vinkel endring for akselerasjon (d^2f/dx^2)
    for j=1:length(w_b_t)-1
         Vinkelakselerasjon(j) = w_b_t(j)-w_b_t(j+1);
    end

    Vinkel = [Vinkel zeros(1,1)];
    Vinkel = [zeros(1,length(Tid)); zeros(1,length(Tid)); Vinkel];

    w_b_t = [w_b_t zeros(1,2)];
    w_b_t = [zeros(1,length(Tid)); zeros(1,length(Tid)); w_b_t];

    p_b_t = [p_b_t(:,1) p_b_t(:,2) ones(length(Tid),1)*1000]';


    for m=1:length(p_b_t(1,:))-1
         v_b_tx(m) = (p_b_t(1,m+1)-p_b_t(1,m))/Ts;
         v_b_ty(m) = (p_b_t(2,m+1)-p_b_t(2,m))/Ts;
    end

    v_b_tx = [v_b_tx ones(1,1)*231];
    v_b_ty = [v_b_ty zeros(1,1)];
    v_b_t = [v_b_tx; v_b_ty; zeros(1,length(Tid))];

    for g=1:length(p_b_t(1,:))-1
        a_b_tx(g) = (v_b_t(1,g+1)-v_b_t(1,g))/Ts;
        a_b_ty(g) = (v_b_t(2,g+1)-v_b_t(2,g))/Ts;
    end

    a_b_tx = [a_b_tx zeros(1,1)];
    a_b_ty = [a_b_ty zeros(1,1)];
    a_b_t = [a_b_tx; a_b_ty; zeros(1,length(Tid))];

    x = [p_b_t; v_b_t; Vinkel];

end  %END FUNCTION
```

C.4 Simulering.m

```
function [f_b,w_b_ib,u] = Simulering(p_b_t,Tid,v_b_t,a_b_t,Vinkel,w_b_t)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               MASTEROPPGAVE               %
%               UNIK Vår 2011               %
%                                           %
%           Funksjon - SIMULERING           %
%                                           %
%        Kenneth Holterhuset Johansen       %
%                 ELD 5930                  %
%                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Initialisering

% Beregning av delta T
    for j = 1:length(601)

        deltaT = Tid(j+1) - Tid(j);

    end

% N brukes til å angi lengde på for-løkker
N = Tid/deltaT;

% g-kraft {t}-ramma
g = 9.81;
g_t = [0;0;-g];

% Jord radien + posisjon {t}-ramma
r_e = 6.370*10^6;
p_t_e = [(r_e*cos(pi/3));
          0;
         (r_e*sin(pi/3))];

% Lengdegrad med RKM R_t_e
L = (pi/3);

Rt = [1 0 0;
      0 cos((pi/2)-(L)) -sin((pi/2)-(L));
      0 sin((pi/2)-(L)) cos((pi/2)-(L))];

Re = [cos(pi/2) -sin(pi/2) 0;
      sin(pi/2) cos(pi/2) 0
      0 0 1];

R_t_e = Re*Rt;

% Vinkelhastigheten til {e}-ramma med RKM R_e_i med hensyn på tiden
w_e = 7.292115*10^-5;
w_e_i = [0;0;w_e];

R_e_i = [1 0 0;
         0 cos(w_e*Tid(1)) -sin(w_e*Tid(1));
         0 sin(w_e*Tid(1)) cos(w_e*Tid(1))];
```

```
% Vinkelhatsighet {t}-ramma, relativt til {e}-ramma
w_t_e = [0;0;0];

%Vinkelx = Vinkel(3,:);
    for k = 1:length(N)

        % Eulervinkler R_b_t = R3*R2*R1 - (kall på funksjon)
        % & transformasjon
        R_b_t = RKMbt(Vinkel(k));
        R_t_b = (R_b_t)';

        % RKM {t} -> {e}
        R_e_t = R_t_e';

        % RKM {b} -> {i} & transformert
        R_b_i = (R_e_i)*(R_t_e)*(R_b_t);
        R_i_b = (R_b_i)';

        % RKM {e} -> {i}
        R_e_i = [cos(w_e*Tid(k)) -sin(w_e*Tid(k)) 0;
                 sin(w_e*Tid(k)) cos(w_e*Tid(k)) 0;
                 0 0 1];

        % RKM {b} -> {e} & transformert
        R_b_e = R_t_e*R_b_t;
        R_e_b = (R_b_e)';

        % Skjevsymmetrisk funksjon
        Sw_e_i = Skjev(w_e_i);

        %% Bruk av likning A-119: [O.Hallingstad, 17.04.08, s.30]
        p_b_i(:,k) = (R_e_i*p_t_e) + (R_e_i*R_t_e*p_b_t(:,k));
        v_b_i(:,k) = (Sw_e_i*R_e_i*p_t_e) +
(Sw_e_i*R_e_i*R_t_e*p_b_t(:,k)) + (R_e_i*R_t_e*v_b_t(:,k));
        a_b_i(:,k) = Sw_e_i*(Sw_e_i*R_e_i*(p_t_e + (R_t_e*p_b_t(:,k))) +
2*R_e_i*R_t_e*v_b_t(:,k)) + R_e_i*R_t_e*a_b_t(:,k);

        %% Vinkelhastighet & Spesifikkkraft (sannverdi)
        w_b_ib(:,k) = (R_i_b*w_e_i) + (R_e_b*w_t_e) + (R_t_b*w_b_t(:,k));
        f_b(:,k) = ((R_i_b*Sw_e_i)*(Sw_e_i*R_e_i*(p_t_e +
(R_t_e*p_b_t(:,k))) + 2*R_e_i*R_t_e*v_b_t(:,k))) + R_t_b*a_b_t(:,k) -
R_t_b*g_t;

    end

    %% Pådragsmatrise u (sannverdi)
    u = [f_b; w_b_ib];

end
```

C.4.1 Skjev.m

```
function Sw_e_i = Skjev(w_e_i)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               MASTEROPPGAVE              %
%               UNIK Vår 2011              %
%                                          %
%             Funksjon - SKJEV             %
%                                          %
%        Kenneth Holterhuset Johansen      %
%                 ELD 5930                 %
%                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Skjevsymmetrisk matrise for w_e_i

Sw_e_i =    [0 -w_e_i(3,:) w_e_i(2,:);
              w_e_i(3,:) 0 -w_e_i(1,:);
              -w_e_i(2,:) w_e_i(1,:) 0];

% Sw_t_i =     [0 -w_t_i(3,:) w_t_i(2,:);
%               w_t_i(3,:) 0 -w_t_i(1,:);
%               -w_t_i(2,:) w_t_i(1,:) 0];

end
```

C.5 NAV.m

```
function xBolge = NAV(x,u,Tid)
%Maintest;
% Initialiseringer
N = length(Tid);
x0 = [x((1:3),1);x((4:6),1);x((7:9),1)];


    for k = 1:N-1
        %% Beregning av navigasjonlikninger

        % Deltaverdi av tiden
        h = Tid(k+1) - Tid(k);

        % Pådraget hentes ved tiden = k
        u(:,k) = [u((1:3),k); u((4:6),k)];
        % Pådraget hentes ved tiden = k+1
        u(:,k+1) = [u((1:3),k+1); u((4:6),k+1)];

        % Euler's metode
        f = funksjon(x0(:,k),u(:,k),Tid(k));
        xm(:,k+1) = x0(:,k) + h*f';

        % Heun's metode
        f2 = funksjon(xm(:,k+1),u(:,k+1),Tid(k+1));
        x0(:,k+1) = x0(:,k) + (h/2)*(f' + f2');

    end

    % x0 = x = Heun
    % Ny variabel, Heun er verdien som blir gitt ut av funksjonen
    xBolge = x0;

end
```

C.5.1 Funksjon.m

```
function f  = funksjon(x,u,Tid)
%% Tilstandene innholdt i x
p_b_t = x(1:3);
v_b_t = x(4:6);
Vinkel = x(7:9);

%% Tilstandene inneholdt i u
f_b = u(1:3);
w_b_ib = u(4:6);

%% Initialisering av konstanter
% g-kraft {t}-ramma
g = 9.81;
g_t = [0;0;-g];

% Jord radien + posisjon {t}-ramma
r_e = 6.370*10^6;
p_t_e = [(r_e*cos(pi/3));
         0;
        (r_e*sin(pi/3))];

% Lengdegrad med RKM R_t_e
L = (pi/3);

Rt = [1 0 0;
      0 cos((pi/2)-(L)) -sin((pi/2)-(L));
      0 sin((pi/2)-(L)) cos((pi/2)-(L))];

Re = [cos(pi/2) -sin(pi/2) 0;
      sin(pi/2) cos(pi/2) 0
      0 0 1];

R_t_e = Re*Rt;
R_e_t = R_t_e';

% Vinkelhastigheten til {e}-ramma med RKM R_e_i med hensyn på tiden
w_e = 7.292115*10^-5;
w_e_i = [0;0;w_e];
R_e_i = [1 0 0;
         0 cos(w_e*Tid) -sin(w_e*Tid);
         0 sin(w_e*Tid) cos(w_e*Tid)];

% Kall på funksjoner: Beregning av Eulervinkler og skjevsymmetrisk
matrise
R_b_t = RKMbt(Vinkel(3));
Sw_e_i = Skjev(w_e_i);
D = DRKM(Vinkel(3));

% Beregning av navigsjonslikningene lagt inn i en f matrise
f(1:3) = v_b_t;
f(4:6) = (((R_b_t*f_b) - (g_t)) -
(R_e_t*(R_e_i)'*Sw_e_i)*(((Sw_e_i*R_e_i)*(p_t_e + (R_t_e*p_b_t))) +
(2*R_e_i*R_t_e*v_b_t)));
f(7:9) = D*(R_b_t*w_b_ib - R_t_e'*R_e_i'*w_e_i);

end
```

C.5.2 DRKM.m

```
function D = DRKM(Vinkel)

% RKM for det kinematiske problem 3-2-1 Eulervinkler
% Likning: (A-106) - [O.Hallingstad, 17.04.08, s. 27]

D = [cos(Vinkel)/cos(0) sin(Vinkel)/cos(0) 0;
     -sin(Vinkel) cos(Vinkel) 0;
     cos(Vinkel)*tan(0) sin(Vinkel)*tan(0) 1];

end
```

C.6 Simuleringsresultater.m

```
function
Simuleringsresultater(p_b_t,v_b_t,a_b_t,f_b,w_b_ib,Vinkel,w_b_t,Vinkelaks
elerasjon,xBolge)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               MASTEROPPGAVE                   %
%               UNIK Vår 2011                   %
%                                               %
%     Funksjon - SIMULERINGSRESULTATER          %
%               plot/figurer                    %
%                                               %
%       Kenneth Holterhuset Johansen            %
%                 ELD 5930                      %
%                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Eulervinkler

figure(2)
subplot(1,1,1)
    hold on
    title('Eulervinkler');
    Vinkelx = plot(Vinkel(1,:),'blue');
    Vinkely = plot(Vinkel(2,:),'green');
    Vinkelz = plot(Vinkel(3,:),'red');

legend([Vinkelx,'\theta_1'],[Vinkely,'\theta_2'],[Vinkelz,'\theta_3']);
    xlabel('sek')
    ylabel('rad')

%% Vinkelhastighet(w_b_tt)

figure(3)
subplot(1,1,1)
    hold on
    title('Vinkelhastigheter - \omega_b^t');
    wbtx = plot(w_b_t(1,:),'blue');
    wbty = plot(w_b_t(2,:),'green');
    wbtz = plot(w_b_t(3,:),'red');
    legend([wbtx,'\omega_1'],[wbty,'\omega_2'],[wbtz,'\omega_3']);
    xlabel('sek')
    ylabel('rad/s')

%% Vinkelakselerasjoner

figure(4)
subplot(1,1,1)
    hold on
    title('Vinkelakselerasjon');
    VinkelAx = plot(Vinkelakselerasjon(1,:),'blue');
    VinkelAy = plot(Vinkelakselerasjon(2,:),'green');
    VinkelAz = plot(Vinkelakselerasjon(3,:),'red');

legend([VinkelAx,'\omega_1*'],[VinkelAy,'\omega_3*'],[VinkelAz,'\omega_3*
']);
    xlabel('sek')
```

```
    ylabel('rad/s^2')

%% Posisjon(p_b_t)

figure(5)
subplot(1,2,1),title('Posisjon - p_b^t');
    hold on
    plot(p_b_t(1,:),p_b_t(2,:));
    axis equal
    xlabel('x[m]')
    ylabel('y[m]')

subplot(1,2,2),title('Posisjon - p_b^t');
    hold on
    pbtx = plot(p_b_t(1,:),'blue');
    pbty = plot(p_b_t(2,:),'green');
    pbtz = plot(p_b_t(3,:),'red');
    legend([pbtx,'x'],[pbty,'y'],[pbtz,'z']);
    axis([0 612 -1 10001])
    xlabel('sek')
    ylabel('meter')

%% Hastighet(v_b_t)

figure(6)
subplot(1,1,1),title('Hastighet - v_b^t');
    hold on
    %plot(v_b_t(1,:),'green');
    vbtx = plot(v_b_t(1,:),'blue');
    vbty = plot(v_b_t(2,:),'green');
    vbtz = plot(v_b_t(3,:),'red');
    legend([vbtx,'x'],[vbty,'y'],[vbtz,'z']);
    axis([0 612 -1 250])
    xlabel('sek')
    ylabel('m/s')

%% Akselerasjon(a_b_t)

figure(7)
subplot(1,1,1),title('Akselerasjon - a_b^t');
    hold on
    abtx = plot(a_b_t(1,:),'blue');
    abty = plot(a_b_t(2,:),'green');
    abtz = plot(a_b_t(3,:),'red');
    legend([abtx,'x'],[abty,'y'],[abtz,'z']);
    axis ([0 612 -60 60])
    xlabel('sek')
    ylabel('m/s^2')

%% Spesifikk kraft (f_b)

figure(8)
subplot(1,1,1)
    hold on
    title('Spesifikk kraft - f^b');
    fbx = plot(f_b(1,:),'blue');
    fby = plot(f_b(2,:),'green');
    fbz = plot(f_b(3,:),'red');
    legend([fbx,'x'],[fby,'y'],[fbz,'z']);
```

```
    axis ([0 612 -60 60])
    xlabel('sek')
    ylabel('m/s^2')

%% Vinkelhastighet (w_b_ib)

figure(9)
subplot(1,1,1)
    hold on
    title('Vinkelhastighet - w_b^i^b');
    wbibx = plot(w_b_ib(1,:),'blue');
    wbiby = plot(w_b_ib(2,:),'green');
    wbibz = plot(w_b_ib(3,:),'red');
    legend([wbibx,'x'],[wbiby,'y'],[wbibz,'z']);
    xlabel('sek')
    ylabel('rad/s')

%% Posisjon (p_b_t) VS Posisjon (NAV)

figure(10)
subplot(2,2,1),title('Posisjon - p_b^t');
    hold on
    plot(p_b_t(1,:),p_b_t(2,:));
    xlabel('x[m]')
    ylabel('y[m]')
    axis equal

subplot(2,2,2),title('Posisjon - p_b^t');
    hold on
    pbtx = plot(p_b_t(1,:),'blue');
    pbty = plot(p_b_t(2,:),'green');
    pbtz = plot(p_b_t(3,:),'red');
    legend([pbtx,'x'],[pbty,'y'],[pbtz,'z']);
    axis([0 612 -1 10001])
    xlabel('sek')
    ylabel('meter')

subplot(2,2,3)
    hold on
    title('Posisjon (NAV)');
    plot(xBolge(1,:),xBolge(2,:));
    axis equal
    xlabel('x[m]')
    ylabel('y[m]')

subplot(2,2,4)
    hold on
    title('Posisjon (NAV)');
    dpbtx = plot(xBolge(1,:),'blue');
    dpbty = plot(xBolge(2,:),'green');
    dpbtz = plot(xBolge(3,:),'red');
    legend([dpbtx,'x'],[dpbty,'y'],[dpbtz,'z']);
    axis([0 612 -1 10001])
    xlabel('sek')
    ylabel('meter')

%% Hastighet (v_b_t) VS Hastighet (NAV)

figure(11)
```

```
subplot(2,1,1)
    hold on
    title('Hastighet (v_b^t)')
    vbtx = plot(v_b_t(1,:),'blue');
    vbty = plot(v_b_t(2,:),'green');
    vbtz = plot(v_b_t(3,:),'red');
    legend([vbtx,'x'],[vbty,'y'],[vbtz,'z']);
    axis([0 612 -1 270])
    xlabel('sek')
    ylabel('m/s')

subplot(2,1,2)
    hold on
    title('Hastighet (NAV)');
    dvbtx = plot(xBolge(4,:),'blue');
    dvbty = plot(xBolge(5,:),'green');
    dvbtz = plot(xBolge(6,:),'red');
    legend([dvbtx,'x'],[dvbty,'y'],[dvbtz,'z']);
    axis ([0 612 -1 270])
    xlabel('sek')
    ylabel('m/s')

%% Orientering (Eulervinkler) VS Orientering (NAV)

figure(12)
subplot(2,1,1)
    hold on
    title('Orientering (Eulervinkler)');
    Vinkelx = plot(Vinkel(1,:),'blue');
    Vinkely = plot(Vinkel(2,:),'green');
    Vinkelz = plot(Vinkel(3,:),'red');

legend([Vinkelx,'\theta_1'],[Vinkely,'\theta_2'],[Vinkelz,'\theta_3']);
    xlabel('sek')
    ylabel('rad')

subplot(2,1,2)
    hold on
    title('Orientering (NAV)');
    dtettax = plot(xBolge(7,:),'blue');
    dtettay = plot(xBolge(8,:),'green');
    dtettaz = plot(xBolge(9,:),'red');
    legend([dtettax,'x'],[dtettay,'y'],[dtettaz,'z']);
    xlabel('sek')
    ylabel('grader')

end
```